▼ AMSTRAD ▼ SPECTRUM ▼ COMMODORE ▼ IBM ▼ MSX ▼

Informática y programación

26

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Informática y programación 26

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.
Gerente:
ANTONIO G. CUERPO.
Directora de producción:
MARIA LUISA SUAREZ PEREZ.
Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática.
JOSE ARTECHE, Ingeniero de Telecomunicación.
Diseño y maquetación:
BRAVO-LOFISH.
Fotografía:
EQUIPO GALATA.
Dibujos:
JOSE OCHOA

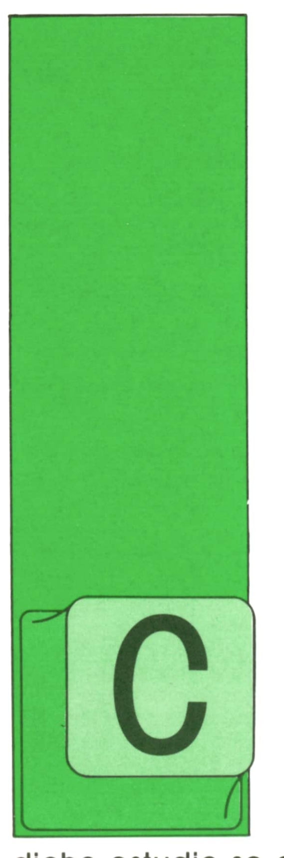
TECNICAS DE PROGRAMACION: Manuel Alfonseca, Doctor Ingeniero de Telecomunicación y Licenciado en Informática. TECNICAS DE ANALISIS: José Arteche, Ingeniero en Telecomunicación. LENGUAJE MAQUINA 8086: Juan Rojas, Licenciado en Ciencias Físicas e Ingeniero Industrial. PASCAL: Juan Ignacio Puyol, Ingeniero Industrial. PROGRAMAS (educativos, de utilidad, de gestión y de juegos): Francisco Morales, Técnico en Informática y colaboradores. Coordinador de Aula de Informática Aplicada (AIA): Alejandro Marcos, Licenciado en Ciencias Químicas. BASIC: Esther Maldonado, Diplomada en Arquitectura. INFORMATICA BASICA: Virginia Muñoz, Diplomada en Informática. LENGUAJE MAQUINA Z-80: Joaquín Salvachúa, Diplomado en Telecomunicación y José Luis Tojo, Diplomado en Telecomunicación. LENGUAJE MAQUINA 6502: (desde el tomo 5): Juan José Gómez, Licenciado en Química. LOGO: Cristina Manzanera, Licenciada en Informática. APLICACIONES: Sociedad Tamariz, Diplomada en Telecomunicación. OTROS LENGUAJES (COBOL): Eloy Pérez, Licenciado en Informática. Ana Pastor, Licenciada en Informática.

Ediciones Siglo Cultural, S.A.
Dirección, redacción y administración:
Pedro Teixeira, 8, 2.ª planta. Teléf. 455 09 99. 28020 Madrid.
Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.
Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.
Distribución en Ecuador: Muñoz Hnos.
Distribución en Perú: DISELPESA.
Distribución en Chile: Alfa Ltda.
Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-167-3
ISBN de la obra: 84-7688-068-7
Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.
Imprime:
MATEU CROMO. Pinto (Madrid).

Depósito legal: M. 5.677-1987
Printed in Spain - Impreso en España.
Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Pedro Teixeira, 8, 2.ª planta. Teléf. 259 73 31. 28020 Madrid.
Noviembre, 1987.
P.V.P. Canarias: 335,–.

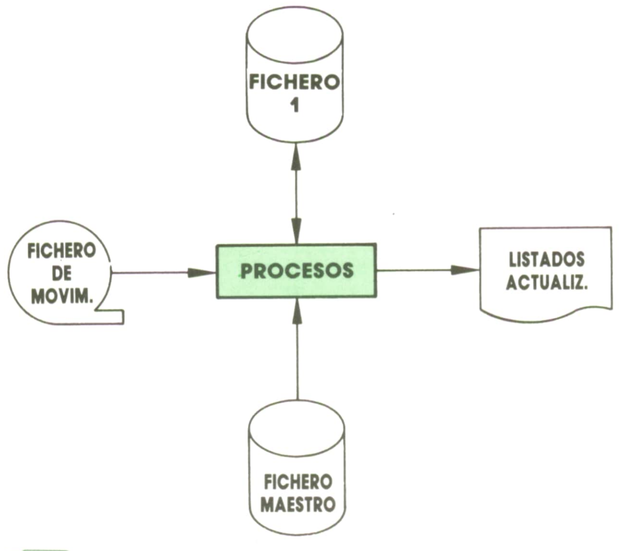
INDICE



INFORMATICA BASICA

FASES DE LA MECANIZACION

CUANDO una empresa decide implantar un sistema informático es necesario que realice un estudio profundo sobre sus necesidades y posibilidades. Una vez hecho dicho estudio se elige el sistema, es decir, el ordenador más indicado junto con sus periféricos, paquetes de software y demás material. Una vez puestos todos los medios comienza la mecanización de «trabajos».



Representación de las tareas a realizar en un proyecto determinado.

Un proyecto informático, no necesariamente muy grande, comprende diversas tareas que, aunque no son fielmente seguidas por todos los analistas, suelen agruparse en las siguientes fases:

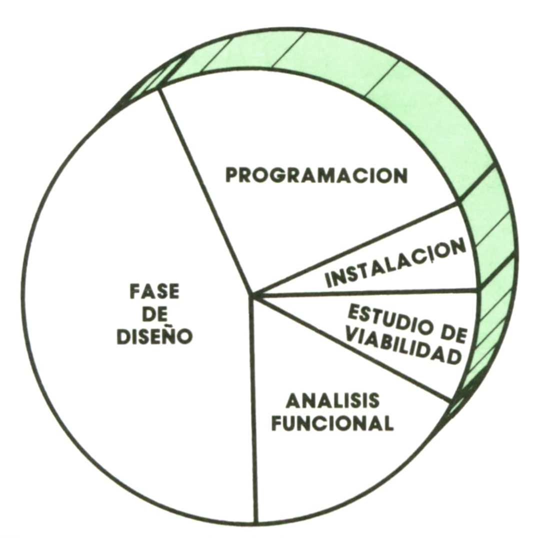
— Estudio de viabilidad.
— Análisis funcional.
— Análisis orgánico.
— Programación.
— Instalación y explotación.

Existen, como en todo, detractores de la realización de los análisis informáticos debido a la cantidad de pasos intermedios y controles que se realizan, sin embargo, teniendo en cuenta la complejidad que suele conllevar un proyecto informático, el «tiempo perdido» en dichos controles asegurarán los objetivos y reducirán los errores de forma bastante eficaz.

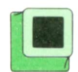
Todas estas fases deben seguir un orden secuencial, ya que cada una se basa en la anterior, resultando que en algunos casos sólo es necesario realizar una traducción de alguna fase realizada previamente.

Aunque todas las etapas son igualmente importantes, en las que más tiempo se invierte, a la hora de elaborar un proyecto informático, son las de diseño (análisis orgánico) y programación.

Todas y cada una de estas fases deben ir acompañadas de una buena documentación en la que se especifique claramente cuáles son sus objetivos y las herramientas utilizadas, teniendo en cuenta que el proyecto lo van a realizar distintos equipos de personal informático y no debe darse pie a que surjan distintas interpretaciones que hagan fracasar los resultados.




Representación de las proporciones del tiempo invertido en cada una de las fases de un proyecto.



Estudio de la viabilidad

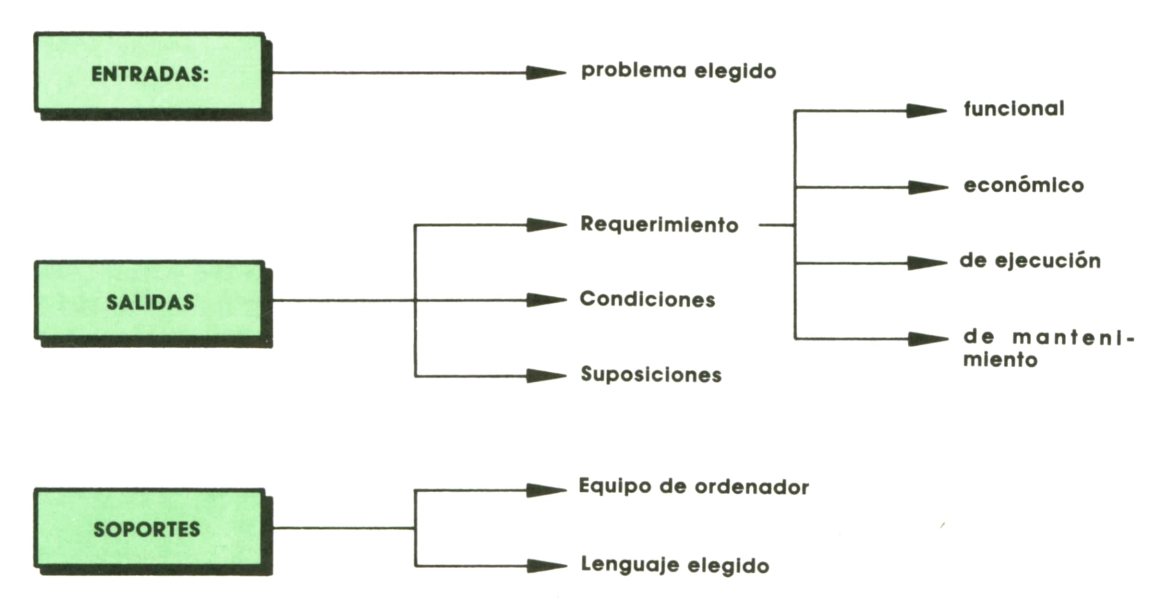
En esta primera fase se estudia si el proyecto objeto de la informatización puede ser adaptado al ordenador o no. Se realiza un estudio detallado del costo de la mecanización, a nivel económico, de tiempo y de personal y se prevén los posibles cambios en la estructura administrativa.

Los resultados en esta fase serán decisivos para afrontar su desarrollo o abandonar el proyecto, ya que ofrecerán los datos que determinen su conveniencia y el costo de la mecanización, además de fijar la fecha en la que la mecanización puede ser finalizada.

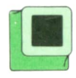
El estudio de los costos y beneficios se debe centrar en dos puntos de vista: físico-lógico y físico.

El físico-lógico se refiere al sistema operativo, lenguajes de programación, rutinas de utilidad, etc.

El estudio físico sería el relativo a la capacidad de memoria central y secundaria, tipo de memoria auxiliar, tipos de terminales, cantidad de terminales, etc.




Fase de especificación.



Análisis funcional

Comienza con la recogida de informaciones: qué es lo que se desea obtener como resultado y a partir de qué.

Es necesario, en este punto, que el usuario especifique exactamente las características de la aplicación que requiere. El analista se encargará de ver cuáles de estas informaciones recibidas del usuario se deben utilizar en el proyecto y cuáles se pueden eliminar.

Una vez recogidos los datos se procede a un análisis detallado de los mismos, tanto de forma cualitativa como cuantitativa.

Toda la labor a realizar se plasma en un lenguaje coloquial, sin demasiados tecnicismos.



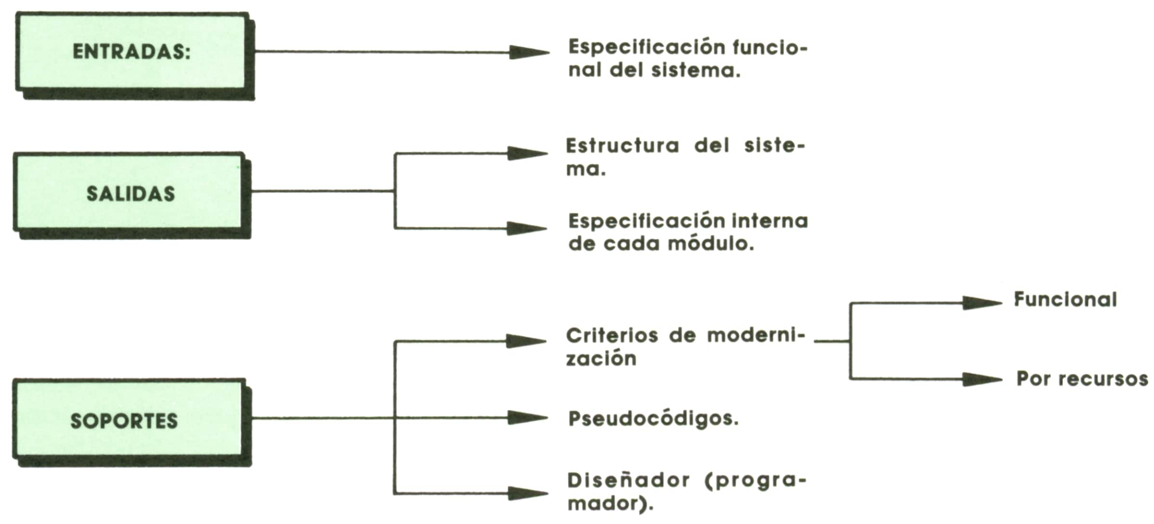
Análisis orgánico

Esta es la fase de diseño. Este punto del proyecto consiste en hacer una «traducción» de la especificación funcional.

Se trata de buscar el tipo de programas y módulos más indicados para el caso. Se estudian los *procesos* que han de efectuarse para que mediante la información de entrada y ficheros de datos especificados en la fase anterior se puedan proporcionar los resultados pedidos.

Esta es labor del *diseñador del sistema* que utiliza, para realizarla, diversas técnicas, como son los seudocódigos (técnicas seminarrativas para describir detalladamente las operaciones a realizar), organigramas y tablas de decisión.

En el diseño tecnológico se trata de determinar los recursos, de entre los elegidos en el plan de mecanización, que habrán de utilizarse para conseguir que el sistema trabaje como se pretende.





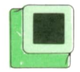
Fase de diseño.

Los recursos tecnológicos son de dos tipos: informáticos y humanos.

El diseño tiene tres fases:

— Diseño de la aplicación.
— Diseño de las cadenas.
— Diseño de los programas.

En cada una de estas fases se considera incluida la codificación y prueba de partes ya diseñadas.

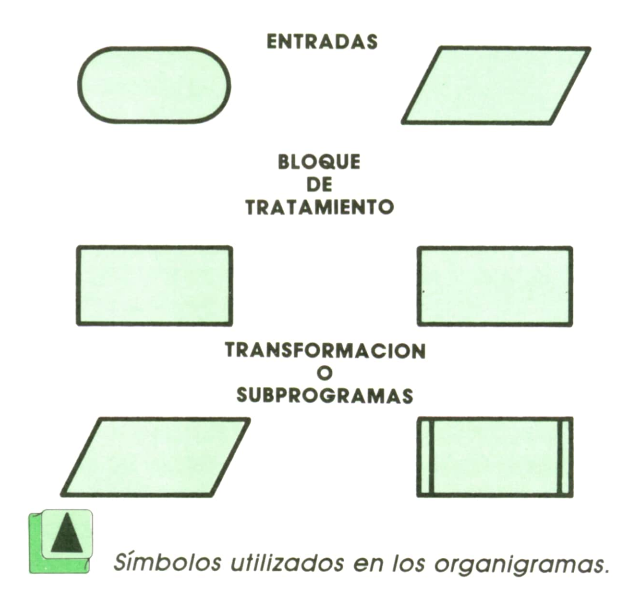


Programación

Es en esta fase cuando se traducen al lenguaje de programación elegido todas las especificaciones de las anteriores etapas.

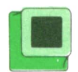
La programación se suele realizar por pasos, es decir, no se hace un único programa, sino varios, comenzando por los más sencillos, que se van probando y combinando luego en grupos más complejos. Por tanto, esta agrupación se conoce con el nombre de «consolidación».

Por supuesto, ésta es labor de los programadores, que finalizará una vez probado todo el sistema, emulando el funcionamiento real.



Símbolos utilizados en los organigramas.

Si se producen errores no se podrá pasar a la fase de instalación hasta no haber corregido todos. En la fase de prueba de los programas se habrá preparado un juego de ensayo (conjunto de datos de entrada que cubre todas las posibilidades), mediante el cual es posible detectar todos los errores lógicos del programa.



Instalación y explotación

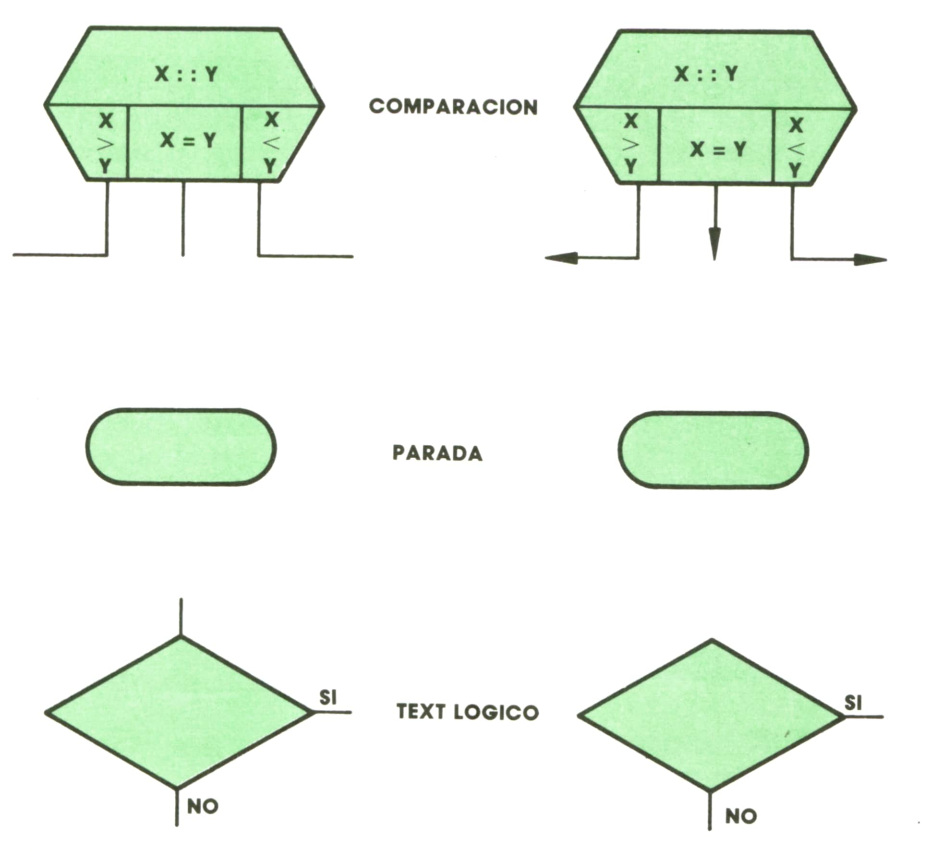
El nuevo sistema, una vez probado, se instala, pero generalmente no de forma definitiva. Suele haber un período de tiempo de prueba en el que trabajan «en paralelo» el sistema antiguo y éste para poder detectar los posibles fallos, no previstos en los juegos de ensayo.

Una vez instalado el sistema comienza la fase de explotación, en la que los programas ejecutan su trabajo normalmente ya con datos reales.

Se deben revisar periódicamente los resultados para comprobar si no se producen fallos y si la nueva mecanización «valió la pena».

En esta fase trabajan equipos de distinta índole:

- Preparadores de trabajos.
- Operadores de consola.
- Controladores de resultados.





Símbolos utilizados en organigramas.

El mantenimiento de las aplicaciones también es un punto importante, ya que consiste en arreglar o innovar programas de forma lo más transparente posible.

Los analistas de mantenimiento deben ser personas muy aventajadas y conocedoras de múltiples aplicaciones.

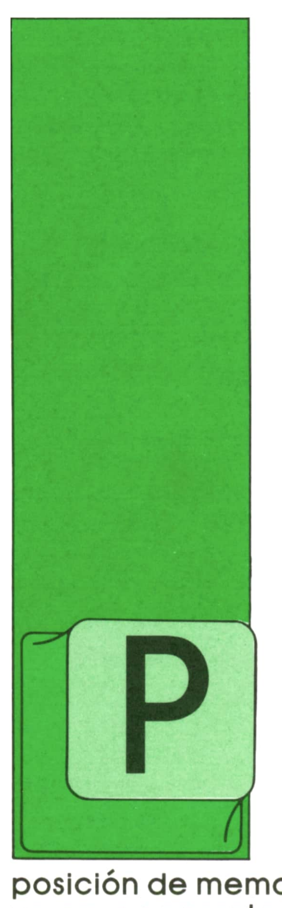
La documentación en la fase de explotación sirve para saber qué datos están contenidos en el sistema de tres formas:

- Como catálogo de programas.
- Como biblioteca de ficheros.
- Como normas de utilización de programas y ficheros.

La documentación en las fases de análisis sirve para decidir cómo incorporar modificaciones al sistema cuando quede desfasado.

MAQUINA 6502

COMANDADOS DE INCREMENTO



PARA poder programar bucles en código máquina el procesador necesita comandos que posibiliten el incremento y decremento, tanto de sus registros como de la posición de memoria que nos interese en un caso concreto. Sería algo parecido al comando NEXT del BASIC.

Son muy sencillos de aprender y también de utilizar. Veámoslo uno a uno.

INY. El contenido del registro Y se incrementa en una unidad. Según el resultado se activarán o desactivarán los flags N y Z.

DEY. El contenido del registro Y se decrementa en una unidad. Según el resultado se modificarán también los flags Z y N.

Los correspondientes comandos para el registro X con análoga aplicación son INX y DEX, respectivamente.

No existe un comando en la serie 65XX para incrementar/decrementar el ACU en una unidad. En casos así, lo que se debe hacer es transferir el valor del ACU al registro X o el Y (TAX o TAY), y una vez aquí, realizar el incremento/decremento.

Para finalizar se vuelve a transferir el valor final al ACU.

INC. Este comando incrementa en uno el contenido de la posición de memoria especificada. Nuevamente se podrán configurar los flags N y Z.

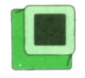
Este es el primer comando que estudiamos en el que se lee un registro, en RAM, y a continuación se sobreescribe en él. El contenido del ACU no se ve alterado por ello.

DEC. Es el comando opuesto al anterior y es válido todo lo dicho para INC, con la excepción de que ahora se decrementa el contenido de una posición de memoria.

A continuación se presenta un resumen de estos comandos con sus códigos.

Comando	Código
INX	$E8
INY	$C8
DEC	$CA
DEY	$88

Modo de direccionamiento	INC	DEC
Absoluto	$EE	$CE
Zeropage	$E6	$C6
Absoluto indexado en X	$FE	$DE
Zeropage indexado en Y	$F6	$D6



Comandos de alteración de flags

Además de la alteración de flags según el resultado de ciertas operaciones, como ya se ha visto, también pueden ser activados o desactivados de manera indirecta.

Esta operación es necesaria antes de una adición o una substracción.

Son comandos de 1 byte y no necesitan, por tanto, operando alguno.

Carry flag

SEC. Se activa el Carry flag debiéndose utilizar antes de una substracción.

CLC. Se desactiva el Carry flag y se debe utilizar antes de una adición.

Decimal flag

SED. Activa el flag y el procesador trabajará en forma decimal.

CLD. Desactiva el flag.

Interrupt flag

SEI. Se activa el flag y el procesador no estará en condición de aceptar una interrupción.

CLI: Se desactiva el flag y las interrupciones vuelven a estar permitidas.

El Overflow flag

Este flag tan sólo puede ser desactivado. Para ello se utiliza el comando CLV.

La tabla siguiente contiene los códigos de los comandos expuestos:

Comando	Código
CLC	$18
SEC	$38
CLD	$D8
SED	$F8
CLI	$58
SEI	$78
CLV	$B8

Bien, ahora, creo que estamos en disposición de crear un programa en lenguaje máquina que haga algo digamos «más útil» que los vistos hasta ahora.

Vamos a intentar una rutina que ponga la pantalla en alta resolución y que vuelva a la modalidad de texto tan sólo pulsando una tecla. Para ello se da por supuesto que el lector posee un «monitor» (programa) que le permite la entrada de programas en assembler.

Si no lo tiene, no se preocupe, intente seguir el proceso que va a estar comentado prácticamente línea por línea, y ya lo tecleará cuando se decida a adquirir el monitor.

Lo primero que vamos a hacer es definir una serie de variables cuyo contenido nos va a hacer falta.

V = 53248 ($D000)
V_1 = V + 17 = 53265 ($D011)
V_2 = V + 24 = 53272 ($D018)
V_3 = 53270 ($D016)

V: Es la dirección inicial del controlador de vídeo, el cual dispone de 47 registros, cuyo significado y características se salen del propósito de este tema.

V_1: El quinto bit de este registro nos indica si estamos en modo gráfico (1) o en modo texto (0).

V_2: Dirección base de la RAM de vídeo (bits 4-7) y del generador de caracteres (bits 1-3).

V_3: El cuarto bit determina el modo multicolor.

```
100    * =$C400
100    ORG=$C400
110    LDA ≠$3B
120    STA $D011
130    LDA ≠$2D
140    STA $D018
150    LDA ≠$D8
160    STA $D016
170    RTS
180    LDA ≠$1B
190    STA $D011
200    LDA ≠$15
210    STA $D018
220    LDA ≠$C8
230    STA $D016
240    RTS
```

Descripción de la rutina:

— La(s) línea(s) 100 son dos de las formas más utilizadas para comunicar el programa ensamblador con el que estamos trabajando, la dirección en la que debe comenzar el ensamblado. Cada ensamblador tiene su propio formato para comunicar el origen de la rutina.

— La posición de memoria 53265 contiene, al encender el ordenador, la configuración %00011011 = $1B = 27. Si ponemos el quinto bit a «1», entraremos en el modo gráfico. %00111011 = $3B = 59. Esta operación se realiza en las líneas 110 y 120.

— La posición de memoria 53272 del controlador de vídeo es muy importante y también conflictiva. Al encender el ordenador su configuración es la siguiente: %00010101 = $1B = 21. Los bits 4-7 definen la posición de la RAM de vídeo. El bit 3 nos va a indicar la posición del mapa de bits. Como sabemos, cada carácter en pantalla se compone de 8 × 8 puntos. Como caben 40 × 25 caracteres, tendremos en total 64.000 puntos en la pantalla de alta resolución, cada uno de los cuales puede estar activado «1» o desactivado «0». Necesitamos un área de memoria de 8.000 bytes (cada byte tiene 8 bits), para almacenar estos datos.

Este área que reservamos se denomina «mapa de bits».

Si el bit 3 de la posición 53272 está a «cero», el mapa de bits comenzará en la posición 0 de memoria, cosa que no nos interesa. Si el bit 3 está a «uno» el mapa de bits empezará en la posición 8192 y terminará en la 16.192. Además podemos desplazar la RAM de vídeo para que no sea borrada al activar la gráfica. Esto se hace modificando los bits 4-7.

BIT 7654	DIRECCION
0000	0
0001	1024
0010	2048
0011	3072
0100	ROM
0101	ROM
0110	ROM
0111	ROM
1000	8192
1001	9216
1010	10240
1011	11264
1100	12288
1110	14336
1111	15360

Esto se hace en las líneas 130 y 140, en las que cargamos el ACU con el número $2D = 45 = %00101101 y lo almacenamos en la posición 53272. Vemos que el bit 3 está activado y el bit 4 ha pasado al 5, es decir, hemos desplazado la RAM de vídeo 1 Kbyte a la posición 2048.

— Por último, el registro número 22, es decir, la posición de memoria 53270, nos determina si está activado el modo multicolor.

La configuración inicial de este registro es %11001000 = $C8 = 200. Al activar el bit cuarto quedará %11011000 = $D8 = $216. Líneas 150 y 160.

El resto de la rutina consiste en volver la pantalla al modo texto, recobrándose los valores iniciales en los registros.

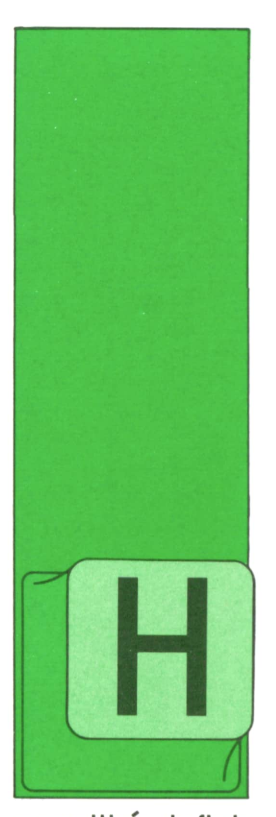
Para comprobar la rutina puede teclear un sencillo programa BASIC que haga a la pantalla ponerse en modo «gráfico» o modo texto al pulsar una tecla.

```
100    SYS 50176:REM ACTIVAR GRAFICA
110    GET A$:IF A$= THEN 110
120    SYS 50192: REM DESACTIVAR GRAFICA
130    GET A$:IF A$="" THEN 130
140    IF A$="F" THEN END
150    GOTO 100
```

PROGRAMAS

EDUCATIVOS • DE UTILIDAD • DE GESTION • DE JUEGOS

Definidos de caracteres para Commodore



HEMOS visto en esta colección una serie de programas para definir SPRITES y caracteres en distintos ordenadores. A continuación vamos a ver un programa que nos permitirá definir nuestro propio juego de caracteres en el COMMODORE.

El uso del programa es muy sencillo. Para mover el cursor por la parrilla de dibujo, donde tendremos que definir los caracteres, debemos utilizar las siguientes teclas:

O: Mover el cursor hacia la derecha.

P: Mover el cursor hacia la izquierda.

Q: Mover el cursor hacia arriba.

A: Mover el cursor hacia abajo.

```
1000 REM ************************************************
1010 REM * DEFINIDOR DE CARACTERES PARA EL COMMODORE 64 *
1020 REM ************************************************
1030 REM
1040 POKE 650,128
1050 POKE 53281 ,254
1060 PRINT CHR$(147)
1070 POKE 56334 ,0
1080 POKE 1,51
1090 FOR K=0 TO 4095
1100    POKE 57344 +K,PEEK(53248 +K)
1110 NEXT K
1120 POKE 1,55
1130 POKE 56334 ,1
1140 FOR I=0 TO 255
1150    POKE 50176 +I,I
1160 NEXT I
1170 DIM C(8),CA%(511,7)
1180 LET PA=1024
1190 POKE 53280 ,1
1200 POKE 53281 ,1
1210 PRINT CHR$(144);CHR$(147)
1220 LET CA=0
1230 LET PO=57344
1240 LET JU=0
1250 GOSUB 1890
1260 GOSUB 2080
1270 LET X=0
1280 LET Y=0
1290 GOSUB 1490
1300 GOSUB 2370
1310 GET A$
1320 IF A$="" THEN GOTO 1310
```

```
1330 LET T=ASC(A$)
1340 IF T<>32 THEN GOTO 1370
1350 IF 2^(7-X) AND C(Y) THEN LET C(Y)=C(Y)-2^(7-X):GOTO 1370
1360 LET C(Y)=C(Y) OR 2^(7-X)
1370 IF T>132 AND T<141 THEN GOTO 1450
1380 LET T=PA+41+X+40*Y
1390 IF C(Y) AND 2^(7-X) THEN POKE T,81:GOTO 1410
1400 POKE T,46
1410 LET X=(X-(A$="P")+(A$="O")) AND 7
1420 LET Y=(Y-(A$="A")+(A$="Q")) AND 7
1430 GOSUB 1640
1440 GOTO 1310
1450 ON T-132 GOSUB 2600,2700,2900,3780,2800,3020,3170,1700
1460 GOTO 1310
1470 REM
1480 REM ********************
1490 REM * ESCRIBE PARRILLA *
1500 REM ********************
1510 REM
1520 PRINT CHR$(19);CHR$(145);
1530 FOR I=0 TO 7
1540    LET T=C(I)
1550    LET T1=128
1560    PRINT CHR$(29);
1570    FOR G=1 TO 8
1580       IF T AND T1 THEN PRINT CHR$(113);:GOTO 1600
1590       PRINT ".";
1600       LET T1=T1/2
1610    NEXT G
1620    PRINT
1630 NEXT I
1640 LET T=PA+41+X+40*Y
1650 IF C(Y) AND 2^(7-X) THEN POKE T,209:RETURN
1660 POKE T,174
1670 RETURN
1680 REM
1690 REM ******************
1700 REM * BORRA PARRILLA *
1710 REM ******************
1720 REM
1730 FOR I=0 TO 7
1740    LET C(I)=0
1750 NEXT I
1760 GOSUB 1490
1770 RETURN
1780 REM
1790 REM ***************************************
1800 REM * CAMBIO DE CHR$ A CODIGO DE PANTALLA *
1810 REM ***************************************
1820 REM
1830 LET T=ASC(A$)
1840 IF T$>"?" AND T<96 THEN LET T=T-64:GOTO 1860
1850 IF T$>CHR$(95) THEN LET T=T-128
1860 RETURN
1870 REM
1880 REM *******************************
1890 REM * PINTA EL MARCO DE LA PARRILLA *
1900 REM *******************************
1910 REM
1920 PRINT CHR$(19);CHR$(111)
1930 FOR I=1 TO 8
1940    PRINT CHR$(163);
1950 NEXT I
1960 PRINT CHR$(112)
1970 FOR I=0 TO 7
1980    PRINT CHR$(116);"        ";CHR$(167)
1990 NEXT I
2000 PRINT CHR$(108);
```

```
2010 FOR I=1 TO 8
2020    PRINT CHR$(114);
2030 NEXT I
2040 PRINT CHR$(186)
2050 RETURN
2060 REM
2070 REM ***********************
2080 REM * PONE MENU PRINCIPAL *
2090 REM ***********************
2100 REM
2110 GOSUB 2460
2120 PRINT CHR$(19);
2130 FOR I=1 TO 13
2140    PRINT CHR$(145);
2150 NEXT I
2160 PRINT "      ------MENU PRINCIPAL------";
2170 PRINT CHR$(145)
2180 PRINT CHR$(18);CHR$(31);"F1";CHR$(146);CHR$(144);
2190 PRINT "-INVIERTE CARACTER. "
2200 PRINT CHR$(18);CHR$(31);"F5;"CHR$(146);CHR$(144);
2210 PRINT "-ESPEJO VERTICAL"
2220 PRINT CHR$(18);CHR$(31);"F2";CHR$(146);CHR$(144);
2230 PRINT "-ROTACION DERECHA.  "
2240 PRINT CHR$(18);CHR$(31);"F6";CHR$(146);CHR$(144);
2250 PRINT "-SAVE/LOAD."
2260 PRINT CHR$(18);CHR$(31);"F3";CHR$(146);CHR$(144);
2270 PRINT "-ROTACION IZQUIERDA."
2280 PRINT CHR$(18);CHR$(31);"F7";CHR$(146);CHR$(144);
2290 PRINT "-JUEGOS/ASIGNA."
2300 PRINT CHR$(18);CHR$(31);"F4";CHR$(146);CHR$(144);
2310 PRINT "-ESPEJO HORIZONTAL. "
2320 PRINT CHR$(18);CHR$(31);"F8";CHR$(146);CHR$(144);
2330 PRINT "-BORRAR."
2340 RETURN
2350 REM
2360 REM **************************
2370 REM * DATOS DE LA DEFINICION *
2380 REM **************************
2390 REM
2400 PRINT CHR$(19);CHR$(145);CHR$(145);CHR$(145);TAB(11);"JUEGO:";JU
2410 PRINT TAB(11);"ULTIMO CARACTER:";CA;CHR$(157);"--";TAB(28);"-";CHR$(31);" "
;CHR$(144)
2420 POKE 1219,CA
2430 RETURN
2440 REM
2450 REM ***********************************
2460 REM * BORRA LAS LINEAS DE LA PANTALLA *
2470 REM ***********************************
2480 REM
2490 FOR X=12 TO 24
2500    POKE 781,X
2510    SYS 59903
2520 NEXT X
2530 PRINT CHR$(19);
2540 FOR G=0 TO 12
2550    PRINT CHR$(17);
2560 NEXT G
2570 RETURN
2580 REM
2590 REM ****************************
2600 REM * INVERSION DE UN CARACTER *
2610 REM ****************************
2620 REM
2630 FOR I=0 TO 7
2640    LET C(I)=ABS(C(I)-255)
2650 NEXT I
2660 GOSUB 1490
```

```
2670 RETURN
2680 REM
2690 REM ********************
2700 REM * ROTACION DERECHA *
2710 REM ********************
2720 REM
2730 FOR I=0 TO 7
2740    LET C(I)=(C(I)*2 AND 255)+(C(I) AND 128)/128
2750 NEXT I
2760 GOSUB 1490
2770 RETURN
2780 REM
2790 REM ********************
2800 REM * ROTACION DERECHA *
2810 REM ********************
2820 REM
2830 FOR I=0 TO 7
2840    LET C(I)=C(I)/2+(C(I) AND 1)*128
2850 NEXT I
2860 GOSUB 1490
2870 RETURN
2880 REM
2890 REM *******************
2900 REM * ESPEJO VERTICAL *
2910 REM *******************
2920 REM
2930 FOR I=0 TO 3
2940    LET T=C(I)
2950    LET C(I)=C(7-I)
2960    LET C(7-I)=T
2970 NEXT I
2980 GOSUB 1490
2990 RETURN
3000 REM
3010 REM *********************
3020 REM * ESPEJO HORIZONTAL *
3030 REM *********************
3040 REM
3050 FOR I-0 TO 7
3060    LET T=C(I)
3070    LET C(I)=0
3080    FOR G=0 TO 3
3090       IF T AND 2^G THEN LET C(I)=C(I)+2^(7-G)
3100       IF T AND 2^(7-G) THEN LET C(I)=C(I)+2^G
3110    NEXT G
3120 NEXT I
3130 GOSUB 1490
3140 RETURN
3150 REM
3160 REM *************
3170 REM * SAVE/LOAD *
3180 REM *************
3190 REM
3200 GOSUB 2460
3210 PRINT TAB(6);"------ MENU DE CASSETTE------"
3220 PRINT
3230 PRINT CHR$(18);CHR$(31);"F1";CHR$(146);CHR$(144);
3240 PRINT "-SAVE JUEGO.";TAB(20);
3250 PRINT CHR$(18);CHR$(31);"F3";CHR$(146);CHR$(144);
3260 PRINT "-LOAD JUEGO."
3270 PRINT CHR$(18);CHR$(31);"F2";CHR$(146);CHR$(144);
3280 PRINT "-RETORNO AL MENU PRINCIPAL."
3290 GET A$
3300 IF A$="" THEN GOTO 3290
3310 LET A=ASC(A$)
3320 IF A=132 THEN LET O$="OPCION SAVE":GOTO 3360
3330 IF A=133 THEN LET O$="OPCION LOAD":GOTO 3360
```

```
3340 IF A=134 THEN GOSUB 2110:RETURN
3350 GOTO 3290
3360 PRINT
3370 PRINT TAB(8);"------";O$;"------"
3380 PRINT
3390 INPUT "NOMBRE ";T$
3400 IF T$="" OR LEN(T$)>10 THEN POKE 781,20:SYS 59903:PRINT CHR$(147):GOTO 3390
3410 REM
3420 REM ********
3430 REM * LOAD *
3440 REM ********
3450 REM
3460 IF A=132 THEN GOTO 3620
3470 PRINT "PULSE ALGUNA TECLA PARA EMPEZAR A CARGAR"
3480 POKE 1,PEEK(1) AND 39
3490 POKE 192,0
3500 GET A$
3510 IF A$="" THEN GOTO 3500
3520 OPEN 1,1,0,T$
3530 FOR I=JU*256 TO JU*256+255
3540    FOR G=0 TO 7
3550       INPUT #1,CA%(I,G)
3560    NEXT G
3570 NEXT I
3580 CLOSE 1
3590 GOTO 3200
3600 REM
3610 REM ********
3620 REM * SAVE *
3630 REM ********
3640 REM
3650 PRINT "PULSE ALGUNA TECLA PARA EMPEZAR A GRABAR"
3660 POKE 1,PEEK(1) AND 39
3670 POKE 192,0
3680 GET A$
3690 IF A$="" THEN GOTO 3680
3700 FOR I=JU*256 TO JU*256+255
3710    FOR G=0 TO 7
3720       PRINT #1,CA%(I,G)
3730    NEXT G
3740 NEXT I
3750 GOTO 3200
3760 REM
3770 REM *****************
3780 REM * JUEGOS/ASIGNA *
3790 REM *****************
3800 REM
3810 GOSUB 2460
3820 PRINT TAB(7);"-------JUEGOS/ASIGNA------"
3830 PRINT
3840 PRINT CHR$(18);CHR$(31);"F1";CHR$(146);CHR$(144);
3850 PRINT "-MENU PRINCIPAL.",
3860 PRINT CHR$(18);CHR$(31);"F4";CHR$(146);CHR$(144);
3870 PRINT "-CAMBIA JUEGO."
3880 PRINT CHR$(18);CHR$(31);"F2";CHR$(146);CHR$(144);
3890 PRINT "-RESUMEN.",
3900 PRINT CHR$(18);CHR$(31);"F5";CHR$(146);CHR$(144);
3910 PRINT "-COPIA DE ROM."
3920 PRINT CHR$(18);CHR$(31);"F3";CHR$(146);CHR$(144);
3930 PRINT "-ASIGNA PARRILLA.",
3940 PRINT CHR$(18);CHR$(31);"F6";CHR$(146);CHR$(144);
3950 PRINT "-EDITA CARACTER."
3960 PRINT
3970 GET A$
3980 IF A$="" THEN GOTO 3970
3990 LET A=ASC(A$)
4000 IF A=133 THEN GOSUB 2110:RETURN
```

```
4010 IF A=134 THEN GOSUB 4640:GOTO 3810
4020 IF A=135 THEN GOSUB 4210:GOTO 3810
4030 IF A=136 THEN GOSUB 4090:GOTO 3810
4040 IF A=137 THEN GOSUB 4460:GOTO 3810
4050 IF A=138 THEN GOSUB 4330:GOTO 3810
4060 GOTO 3970
4070 REM
4080 REM ****************
4090 REM * CAMBIA JUEGO *
4100 REM ****************
4110 REM
4120 PRINT "JUEGO (0-1) ? ";
4130 GET A$
4140 IF A$="" OR A$<>"1" AND A$<>"0" THEN GOTO 4130
4150 LET JU=VAL(A$)
4160 PRINT A$
4170 GOSUB 2370
4180 RETURN
4190 REM
4200 REM *******************
4210 REM * ASIGNA PARRILLA *
4220 REM *******************
4230 REM
4240 GOSUB 4780
4250 LET D=JU*2048+CA*8+57344!
4260 FOR G=0 TO 7
4270    POKE D+G,C(G)
4280    LET CA%(JU*256+CA,G)=C(G)
4290 NEXT G
4300 RETURN
4310 REM
4320 REM ******************
4330 REM * EDITA CARACTER *
4340 REM ******************
4350 REM
4360 GOSUB 4780
4370 FOR G=0 TO 7
4380    LET C(G)=CA%(JU*256+CA,G)
4390 NEXT G
4400 GOSUB 1490
4410 LET X=0
4420 LET Y=0
4430 RETURN
4440 REM
4450 REM ****************
4460 REM * COPIA DE ROM *
4470 REM ****************
4480 REM
4490 GOSUB 4780
4500 POKE 56334 ,0
4510 POKE 1,51
4520 LET D=JU*2048+CA*8+53248
4530 FOR G=0 TO 7
4540    LET C(G)=PEEK(D+G)
4550 NEXT G
4560 POKE 1,55
4570 POKE 56334 ,1
4580 GOSUB 1490
4590 LET X=0
4600 LET Y=0
4610 RETURN
4620 REM
4630 REM ********************************************************
4640 REM * RESUMEN DE CARACTERES CON LOS QUE SE ESTA TRABAJANDO *
4650 REM ********************************************************
4660 REM
4670 POKE 56576 ,196
```

```
4680 POKE 648,196
4690 POKE 53272 ,PEEK(53272 ) AND NOT 6 OR 8
4700 GET A$
4710 IF A$="" THEN GOTO 4700
4720 POKE 53272 ,21
4730 POKE 648,4
4740 POKE 56576 ,199
4750 RETURN
4760 REM
4770 REM *****************
4780 REM * INPUT CARACTER *
4790 REM *****************
4800 REM
4810 PRINT "INTRODUZCA CARACTER:0-NORMAL,1-INVERSO"
4820 GET A$
4830 IF A$="" OR A$<>"0" OR A$<>"1" THEN GOTO 4820
4840 LET T1=VAL(A$)
4850 PRINT "CARACTER ? ";
4860 GET T$
4870 IF T$="" THEN GOTO 4860
4880 PRINT T$
4890 GOSUB 1800
4900 LET CA=T1*128+T
4910 GOSUB 2370
4920 RETURN
```

Una vez que nos encontremos sobre el punto al cual queremos cambiar de estado, pulsamos la barra espaciadora. Si dicho punto estaba apagado, se encenderá, y si estaba encendido, se apagará.

En la parte inferior de la pantalla, podemos ver un menú con todas las posibles opciones que nos permite el programa. Estas son:

F1. INVIERTE CARACTER. Todos los puntos que están iluminados se apagan y todos los que están apagados se iluminan.

F2. ROTACION DERECHA. El carácter rota un punto hacia la derecha. Lo que desaparezca por ésta, vuelve a aparecer por la izquierda.

F3. ROTACION IZQUIERDA. Lo mismo que F2 rota hacia la izquierda.

F4. ESPEJO HORIZONTAL. Nos muestra el carácter como si lo estuviésemos viendo en un espejo.

F5. ESPEJO VERTICAL. Realiza el mismo efecto que F4, pero verticalmente.

F6. SAVE/LOAD. Nos permite leer del cassette o graba en el cassette el juego de caracteres que estamos redifiniendo.

F7. JUEGOS/ASIGNA. Nos muestra otro menú.

F8. BORRAR. Nos borra la parrilla de dibujo.

Si pulsamos la tecla F7 nos aparecerá otro menú con distintas opciones:

F1. MENU PRINCIPAL. Nos devuelve al menú principal.

F2. RESUMEN. Nos permite ver los caracteres que hemos definido hasta el momento.

F3. ASIGNA, PARRILLA. Asigna la figura que está en la parrilla a un carácter en especial.

F4. CAMBIA JUEGO. Nos permite cambiar el juego de caracteres sobre el que estamos trabajando.

F5. COPIA DE ROM. Copia un carácter de la ROM en la parrilla.

F6. EDITA CARACTER. Edita uno de los caracteres del juego en el que nos encontramos.



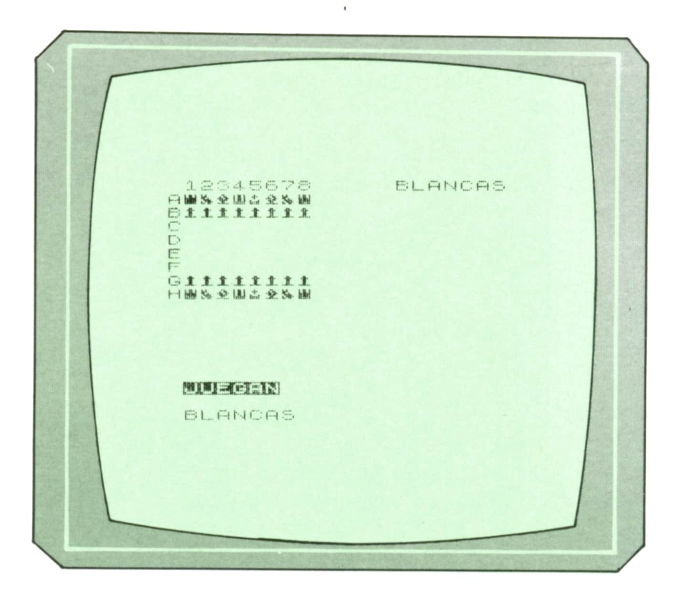
Programa: Ajedrez para Spectrum

El programa de ajedrez que ofrecemos a continuación está pensado para dos jugadores. Esto significa que el ordenador no puede jugar. Sólo se encarga de mover las figuras y de comprobar que los movimientos que se realizan son correctos.

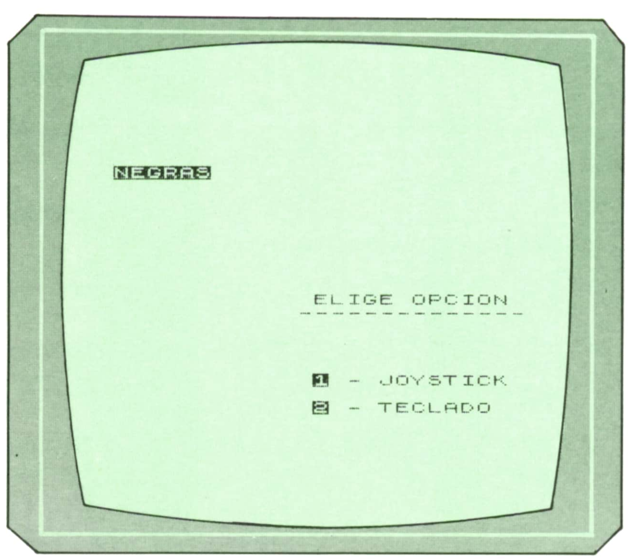


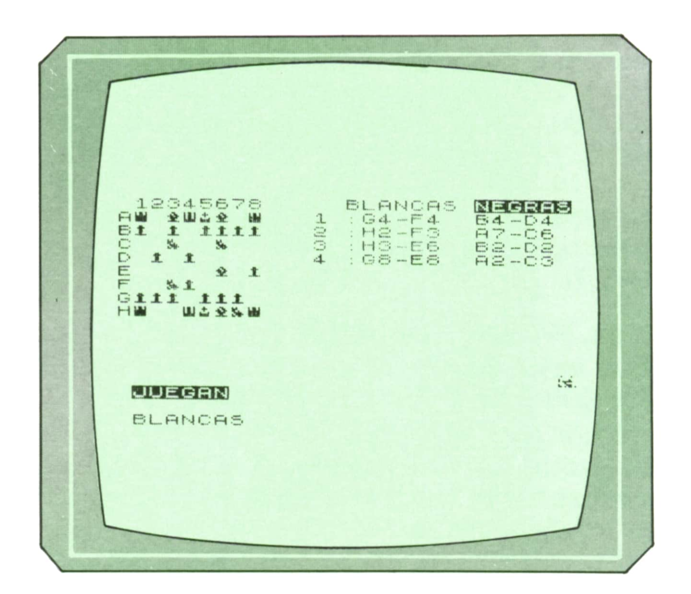
Al principio del programa se pregunta si se quiere jugar con JOYSTICK o con el teclado. Si elegimos la palanca de juegos, ésta tiene que ser compatible KEMSTON.

En el caso de que utilicemos el teclado, para decir el lugar donde queremos mover una de las piezas, tenemos que dar primero las coordenadas de partida, y después, las de destino. Primero daremos la letra, y después, el número.




Comienzo del juego.






Menú de opciones.

```
10 REM ***********************
20 REM ***********************
30 REM *    A J E D R E Z    *
40 REM ***********************
```

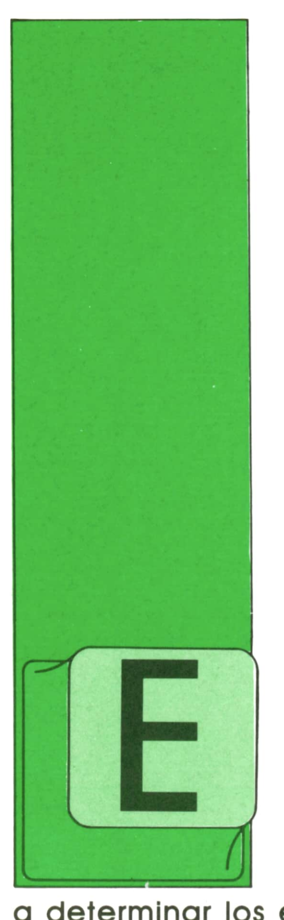
```
  50 REM ***********************
  60 REM * JOSE M.GARCIA LUENGO*
  70 REM ***********************
  80 REM ***********************
  81 REM
  82 REM ******************************
  83 REM *                            *
  84 REM * (c) Ediciones Siglo Cultural *
  85 REM *                            *
  86 REM * (c) 1987                   *
  87 REM *                            *
  88 REM ******************************
  89 REM
  90 PAPER 0
 100 INK 6
 110 BORDER 0
 120 CLEAR
 130 PRINT AT 2,10; INK 4;"ELIGE OPCION";AT 3,9; INK 5;"--------------"
 140 PRINT AT 8,10; INK 7; INVERSE 1; PAPER 1;"1"; INVERSE 0; PAPER 0;" - JOYSTI
CK"
 145 PRINT AT 10,10; INK 7; INVERSE 1; PAPER 1;"2"; INVERSE 0; PAPER 0;" - TECLA
DO"
 150 IF INKEY$="1" THEN  LET JOY=1: GO TO 180
 160 IF INKEY$<>"2" THEN  GO TO 150
 170 LET JOY=0
 171 REM
 172 REM *****************
 173 REM * INICIALIZACION *
 174 REM *****************
 175 REM
 180 CLS
 181 PRINT AT 10,8; FLASH 1;"ESPERE UN MOMENTO"
 190 GO SUB 1870
 200 LET ERR=0
 210 LET NJ=1
 220 DIM T$(8,24)
 230 LET T=7
 240 LET PAG=1
 250 LET RLB=0
 255 LET RCB=0
 260 LET RLN=0
 265 LET RCN=0
 270 IF JOY=1 THEN  LET ST=0: LET XJ=5: LET YJ=5
 280 FOR A=1 TO 8
 290    FOR B=3 TO 24 STEP 3
 300       LET T$(A,B)=("4" AND (A+B)/2=INT ((A+B)/2))+("2" AND (A+B)/2<>INT ((A
+B)/2))
 310    NEXT B
 320 NEXT A
 330 FOR A=1 TO 8
 340    FOR B=2 TO 24 STEP 3
 350       LET T$(A,B)=("0" AND A<3)+("7" AND A>6)+("8" AND A>2 AND A<7)
 360    NEXT B
 370 NEXT A
 380 FOR A=1 TO 24 STEP 3
 390    LET T$(2,A)=CHR$ 144
 400    LET T$(7,A)=CHR$ 144
 410 NEXT A
 420 LET T$(1,1)=CHR$ 145
 422 LET T$(1,22)=CHR$ 145
 424 LET T$(8,1)=CHR$ 145
 426 LET T$(8,22)=CHR$ 145
 430 LET T$(1,4)=CHR$ 146
 432 LET T$(1,19)=CHR$ 146
 434 LET T$(8,4)=CHR$ 146
 436 LET T$(8,19)=CHR$ 146
 440 LET T$(1,7)=CHR$ 147
 442 LET T$(1,16)=CHR$ 147
```

```
 444 LET T$(8,7)=CHR$ 147
 446 LET T$(8,16)=CHR$ 147
 450 LET T$(1,10)=CHR$ 149
 452 LET T$(8,10)=CHR$ 149
 458 LET T$(1,13)=CHR$ 148
 459 LET T$(8,13)=CHR$ 148
 461 REM
 462 REM **********************
 463 REM * PRÓGRAMA PRINCIPAL *
 464 REM **********************
 465 REM
 469 CLS
 470 FOR A=1 TO 8
 480    PRINT AT 0,A;A
 490    PRINT AT A,0;CHR$ (64+A)
 500 NEXT A
 510 FOR A=1 TO 8
 520    FOR B=1 TO 24 STEP 3
 530       PRINT  BRIGHT 1; OVER 1;AT A,B/3+1; INK VAL T$(A,B+1); PAPER VAL T$(A
 B+2);T$(A,B)
 540    NEXT B
 550 NEXT A
 560 PRINT  BRIGHT 1;AT 0,14;"BLANCAS";AT 0,22; INVERSE 1;"NEGRAS"; FLASH 1;AT 1
5,1;"JUEGAN"
 570 PRINT AT 21,0;"                              "
 580 PRINT AT 17,1; FLASH 1;("BLANCAS" AND T=7)+("NEGRAS." AND T=0)
 590 IF JOY=1 THEN  GO TO 1930
 600 INPUT "JUGADA:"; LINE C$
 610 IF LEN C$<>5 THEN  GO TO 570
 620 IF C$(1)<"A" OR C$(1)>"H" OR C$(2)>"8" OR C$(2)<"1" OR C$(3)<>"-" OR C$(4)<
"A" OR C$(4)>"H" OR C$(5)>"8" OR C$(5)<"1" THEN  GO TO 570
 630 LET X=VAL C$(2)
 640 LET XF=VAL C$(5)
 650 LET Y=CODE C$(1)-64
 660 LET YF=CODE C$(4)-64
 670 GO SUB 700
 680 IF ERR=1 THEN  LET ERR=0: GO TO 570
 690 GO TO 780
 700 IF VAL T$(Y,X*3-1)=7 THEN  IF T=0 THEN  PRINT AT 21,0;"ESA FICHA NO ES TUYA
...": GO TO 740
 710 IF VAL T$(Y,X*3-1)=0 THEN  IF T=7 THEN  PRINT AT 21,0;"ESA FICHA NO ES TUYA
...": GO TO 740
 720 IF T$(Y,X*3-1)="8" AND T$(Y,X*3-2)=" " THEN  PRINT AT 21,0;"AHI NO HAY NING
UNA FICHA...": GO TO 740
 730 RETURN
 740 BEEP .1,0: PAUSE 35
 750 IF INKEY$="" AND IN 223=0 THEN  GO TO 750
 760 LET ERR=1
 770 RETURN
 780 IF T$(Y,X*3-2)=CHR$ 144 THEN  GO TO 1050: GO TO 2160
 790 IF T$(Y,X*3-2)=CHR$ 149 THEN  GO TO 1180
 800 IF T$(Y,X*3-2)=CHR$ 146 THEN  GO TO 1350
 810 IF T$(Y,X*3-2)=CHR$ 147 THEN  GO TO 1420
 820 IF T$(Y,X*3-2)=CHR$ 149 THEN  GO TO 1500
 830 IF T$(Y,X*3-2)=CHR$ 148 THEN  GO TO 1530
 840 IF T$(Y,X*3-2)=CHR$ 144 THEN  GO SUB 2160
 850 IF T$(YF,XF*3-2)=CHR$ 148 THEN  PRINT  FLASH 1;AT 10,0;"HAN GANADO LAS ";("
BLANCAS" AND T=7)+("NEGRAS" AND T=0): FOR A=-40 TO 40: BEEP .01,A: NEXT A: PAUSE
 0: RUN
 860 IF X=1 AND Y=8 AND T=7 THEN  LET RLB=1
 870 IF X=8 AND Y=8 AND T=7 THEN  LET RCB=1
 880 IF X=1 AND Y=1 AND T=0 THEN  LET RLN=1
 890 IF X=8 AND Y=1 AND T=0 THEN  LET RCN=1
 900 IF T$(Y,X*3-2)=CHR$ 148 AND T=7 THEN  LET RLB=1: LET RCB=1
 910 IF T$(Y,X*3-2)=CHR$ 148 AND T=0 THEN  LET RLN=1: LET RCN=1
 920 LET T$(YF,XF*3-2)=T$(Y,X*3-2)
 930 LET T$(YF,XF*3-1)=T$(Y,X*3-1)
```

```
 940 LET T$(Y,X*3-2)=" "
 950 LET T$(Y,X*3-1)="8"
 960 PRINT AT Y,X; BRIGHT 1; INK VAL T$(Y,X*3-1); PAPER VAL T$(Y,X*3);T$(Y,X*3-2
)
 970 PRINT AT YF,XF; BRIGHT 1; INK VAL T$(YF,XF*3-1); PAPER VAL T$(YF,XF*3);T$(Y
F,XF*3-2)
 980 IF T=7 THEN  PRINT AT NJ-PAG*20+20,12;NJ;AT NJ-PAG*20+20,14;":";C$
 990 IF T=0 THEN  PRINT AT NJ-20*PAG+20,22;C$
1000 LET T=T+(7 AND T=0)-(7 AND T=7)
1010 IF T=7 THEN  LET NJ=NJ+1
1020 IF NJ>20*PAG THEN  LET PAG=PAG+1
1030 GO TO 570
1040 STOP
1041 REM
1042 REM **********************
1050 REM * RESTRICCIONES PEON *
1051 REM **********************
1052 REM
1060 IF ABS (Y-YF)>2 THEN  GO TO 1830
1070 IF ABS (Y-YF)=0 THEN  GO TO 1830
1080 IF T=7 THEN  IF Y-YF<0 THEN  GO TO 1830
1090 IF T=0 THEN  IF YF-Y<0 THEN  GO TO 1830
1100 IF Y-YF=2 THEN  IF T$(YF,XF*3-1)<>"8" OR T$(YF+1,XF*3-1)<>"8" OR Y<>7 OR X-
XF<>0 THEN  GO TO 1830
1110 IF Y-YF=1 AND X=XF THEN  IF T$(YF,XF*3-1)<>"8" THEN  GO TO 1830
1120 IF YF-Y=2 THEN  IF T$(YF,XF*3-1)<>"8" OR T$(YF-1,XF*3-1)<>"8" OR Y<>2 OR X-
XF<>0 THEN  GO TO 1830
1130 IF YF-Y=1 AND X=XF THEN  IF T$(YF,XF*3-1)<>"8" THEN  GO TO 1830
1140 IF ABS (XF-X)>1 THEN  GO TO 1830
1150 IF ABS (XF-X)>1 THEN  GO TO 1830
1160 IF ABS (XF-X)=1 THEN  IF T$(YF,XF*3-2)=" " OR T$(YF,XF*3-1)=STR$ T THEN  GO
 TO 1830
1170 GO TO 840
1171 REM
1172 REM ***********************
1180 REM * RESTRICCIONES TORRE *
1182 REM ***********************
1183 REM
1190 IF YF-Y<>0 THEN  IF XF-X<>0 THEN  GO TO 1830
1200 IF ABS (YF-Y)<>0 THEN  GO TO 1280
1210 IF XF=X+1 OR XF=X-1 THEN  IF T$(YF,XF*3-1)<>STR$ T THEN  GO TO 840
1220 IF X>XF THEN  GO TO 1240
1230 FOR A=X+1 TO XF: GO TO 1250
1240 FOR A=XF TO X-1
1250 IF T$(YF,A*3-1)<>STR$ T OR T$(YF,A*3-1)="8" THEN  NEXT A: GO TO 1270
1260 GO TO 1830
1270 GO TO 840
1280 IF YF=Y+1 OR YF=Y-1 THEN  IF T$(YF,XF*3-1)<>STR$ T THEN  GO TO 840
1290 IF Y>YF THEN  GO TO 1310
1300 FOR A=Y+1 TO YF: GO TO 1320
1310 FOR A=YF TO Y-1
1320 IF T$(A,XF*3-1)<>STR$ T OR T$(A,XF*3-1)="8" THEN  NEXT A: GO TO 1340
1330 GO TO 1830
1340 GO TO 840
1341 REM
1342 REM *************************
1350 REM * RESTRICCIONES CABALLO *
1352 REM *************************
1353 REM
1360 IF T$(Y,X*3)=T$(YF,XF*3) THEN  GO TO 1830
1370 IF T$(YF,XF*3-1)=STR$ T THEN  GO TO 1830
1380 IF ABS (X-XF)>2 OR ABS (Y-YF)>2 OR X-XF=0 OR Y-YF=0 THEN  GO TO 1830
1390 IF ABS (X-XF)=2 THEN  IF ABS (Y-YF)>1 THEN  GO TO 1830
1400 IF ABS (Y-YF)=2 THEN  IF ABS (X-XF)>1 THEN  GO TO 1830
1410 GO TO 840
1411 REM
1412 REM ***********************
```

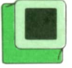
```
1420 REM * RESTRICCIONES ALFIL *
1422 REM ***********************
1423 REM
1430 IF T$(Y,X*3)<>T$(YF,XF*3) THEN  GO TO 1830
1440 IF ABS (X-XF)<>ABS (Y-YF) THEN  GO TO 1830
1450 IF X>XF THEN  GO TO 1470
1460 LET B=Y-(1 AND Y>YF)+(1 AND Y<YF): FOR A=X+1 TO XF: GO TO 1480
1470 LET B=YF: FOR A=XF TO X-1
1480 IF T$(B,A*3-1)<>STR$ T THEN  LET B=B+(1 AND Y<YF AND X<XF)+(1 AND Y>YF AND
X>XF)-(1 AND X>XF AND Y<YF)-(1 AND X<XF AND Y>YF): NEXT A: GO TO 840
1490 GO TO 1830
1491 REM
1492 REM **********************
1500 REM * RESTRICÇIONES DAMA *
1502 REM **********************
1503 REM
1510 IF ABS (X-XF)=ABS (Y-YF) THEN  GO TO 1430
1520 GO TO 1180
1521 REM
1522 REM *********************
1530 REM * RESTRICCIONES REY *
1532 REM *********************
1533 REM
1540 IF Y-YF=0 THEN  IF X-XF=2 AND T=7 AND RLB=0 THEN  GO TO 1610
1550 IF Y-YF=0 THEN  IF X-XF=2 AND T=0 AND RLN=0 THEN  GO TO 1610
1560 IF Y-YF=0 THEN  IF XF-X=2 AND T=7 AND RCB=0 THEN  GO TO 1720
1570 IF Y-YF=0 THEN  IF XF-X=2 AND T=0 AND RCN=0 THEN  GO TO 1720
1580 IF ABS (X-XF)>1 OR ABS (Y-YF)>1 THEN  GO TO 1830
1590 IF T$(YF,XF*3-1)=STR$ T THEN  GO TO 1830
1600 GO TO 840
1601 REM
1602 REM *****************
1610 REM * ENROQUE LARGO *
1612 REM *****************
1613 REM
1620 FOR A=X-1 TO XF+1 STEP -1
1630 IF T$(Y,A*3-2)=" " THEN  NEXT A: GO TO 1650
1640 GO TO 1830
1650 LET T$(Y,10)=T$(Y,1)
1660 LET T$(Y,11)=T$(Y,2)
1670 LET T$(Y,1)=" "
1680 LET T$(Y,2)="8"
1690 PRINT AT Y,4; INK VAL T$(Y,11); PAPER VAL T$(Y,12); BRIGHT 1;T$(Y,10)
1700 PRINT AT Y,1; INK VAL T$(Y,2); PAPER VAL T$(Y,3); BRIGHT 1;T$(Y,1)
1710 GO TO 840
1711 REM
1712 REM *****************
1720 REM * ENROQUE CORTO *
1722 REM *****************
1723 REM
1730 FOR A=X+1 TO XF
1740 IF T$(Y,A*3-2)=" " THEN  NEXT A: GO TO 1760
1750 GO TO 1830
1760 LET T$(Y,16)=T$(Y,22)
1770 LET T$(Y,17)=T$(Y,23)
1780 LET T$(Y,22)=" "
1790 LET T$(Y,23)="8"
1800 PRINT AT Y,6; INK VAL T$(Y,17); PAPER VAL T$(Y,18); BRIGHT 1;T$(Y,16)
1810 PRINT AT Y,8; INK VAL T$(Y,23); PAPER VAL T$(Y,24); BRIGHT 1;T$(Y,22)
1820 GO TO 840
1830 PRINT AT 21,0;"MOVIMIENTO ILEGAL..."
1834 REM
1840 BEEP .1,0: PAUSE 35
1850 IF INKEY$="" AND IN 223=0 THEN  GO TO 1850
1860 GO TO 570
1861 REM
1862 REM ****************************
```

```
1863 REM * DEFINICION DE CARACTERES *
1864 REM ****************************
1865 REM
1870 FOR A=0 TO 47
1880    READ B
1890    POKE A+USR "A",B
1900 NEXT A
1910 RETURN
1920 DATA 0,24,60,24,24,24,126,0,0,90,90
1922 DATA 126,126,126,126,0,0,80,112,104
1924 DATA 60,94,108,0,0,24,44,118,60,24
1926 DATA 126,0,0,16,56,16,68,124,124,0
1928 DATA 0,90,90,90,90,126,126,0
1929 REM
1930 REM ********************
1932 REM * CONTROL JOYSTICK *
1933 REM ********************
1934 REM
1940 LET STICK=IN 223
1950 IF STICK=2 THEN  LET XJ=XJ-(1 AND XJ>1)
1960 IF STICK=1 THEN  LET XJ=XJ+(1 AND XJ<8)
1970 IF STICK=4 THEN  LET YJ=YJ+(1 AND YJ<8)
1980 IF STICK=8 THEN  LET YJ=YJ-(1 AND YJ>1)
1990 IF STICK=16 THEN  IF ST=1 THEN  BEEP .1,40: GO TO 2090
2000 IF STICK=16 THEN  IF ST=0 THEN  BEEP .1,-40: GO SUB 2040
2010 PRINT AT YJ,XJ; INK 7; PAPER VAL T$(YJ,XJ*3); BRIGHT 1;"X"
2020 PRINT AT YJ,XJ; INK VAL T$(YJ,XJ*3-1); PAPER VAL T$(YJ,XJ*3); BRIGHT 1;T$(Y
J,XJ*3-2)
2030 GO TO 1940
2040 LET X=XJ
2050 LET Y=YJ
2060 LET C$=CHR$ (64+Y)+STR$ X+"-"
2070 LET ST=1
2080 RETURN
2090 LET XF=XJ
2100 LET YF=YJ
2110 IF X=XF AND Y=YF THEN  LET ST=0: GO TO 1830
2120 LET C$=C$+CHR$ (64+YF)+STR$ XF
2130 LET ST=0
2140 PRINT AT 20,0;"JUGADA:";C$
2150 GO TO 670
2151 REM
2152 REM *************************
2160 REM * CORONACION DE UN PEON *
2162 REM *************************
2163 REM
2170 IF T=0 THEN  GO TO 2240
2180 IF YF=1 THEN  GO TO 2200
2190 RETURN
2200 INPUT "T/C/A/D ?"; LINE Z$
2210 IF Z$<>"T" AND Z$<>"C" AND Z$<>"A" AND Z$<>"D" THEN  GO TO 2200
2220 LET T$(Y,X*3-2)=(CHR$ 149 AND Z$="T")+(CHR$ 146 AND Z$="C")+(CHR$ 147 AND Z
$="A")+(CHR$ 149 AND Z$="D")
2230 RETURN
2240 IF YF=8 THEN  GO TO 2200
2250 RETURN
```



TECNICAS DE ANALISIS

FASES DE LOS ESTUDIOS INFORMATICOS (II)



ESTUDIO DE VIABILIDAD

ESTE estudio realiza una aproximación global a nivel de un dominio de actividad de la empresa (producción, personal, etc.). Con el estudio de viabilidad se llega a determinar los elementos que han de ser tenidos en cuenta para la definición de las soluciones funcionales y técnicas sobre el dominio considerado dentro del esquema global concebido para el conjunto de la empresa. Se concibe el «plan-masa» de las aplicaciones a desarrollar en ese dominio empresarial con mayor coherencia global y con economía de estudios de detalle.

Normalmente un estudio de viabilidad se desarrolla en cuatro fases sucesivas: una primera de observación, otras dos posteriores de concepción y organización de las soluciones, para concluir con una fase de validación. Veamos con más detalle cada una de estas fases:

a) *Fase de observación.*

Se trata de examinar el área correspondiente hasta su definición conceptual (lo más exhaustiva posible). Hay que realizar una labor sistemática de estructuración y selección de los elementos a considerar, para establecer un adecuado diagnóstico sobre el área considerada.

Al concluir esta fase de observación, todas las personas participantes en ella (los componentes del grupo director del proyecto) han de estar de acuerdo en varios aspectos básicos:

— los límites exactos del campo de visión del estudio que se está desarrollando;

— el subconjunto de este campo de estudio que parece más representativo para la aplicación del «plan-masa» que se va a establecer;

— la definición y valoración de los procedimientos existentes y del sistema de información sobre el que se apoyan, y

— las orientaciones a primar para concebir las hipótesis de las soluciones futuras.

b) *Fase de concepción.*

Se trata de establecer las soluciones conceptuales en el dominio objeto del estudio y su articulación en soluciones lógicas que luego (en la tercera fase) serán desarrolladas y articuladas en detalle.

Es importante sentar las bases para el desarrollo de las soluciones correspondientes antes de abordar esta fase. Es útil, por tanto, que el equipo director decida entre varias posibilidades alternativas, basadas en los modelos conceptuales y lógicos correspondientes, en los siguientes aspectos:

— reglas operativas nuevas, informaciones que van a manejar;

— reparto de carga a efectuar en la ejecución de estas reglas y en la gestión

de la información entre las diferentes unidades geográficas o funcionales (la dirección regional, el servicio de personal, etc.);

— utilización de técnicas específicas, desde el punto de vista informático: tareas conversacionales, tratamientos por lotes, etc.;

— orientaciones a primar en cuanto a puesta a disposición del proyecto, de medios informáticos (telemática, microinformática, software standard, etc.).

c) *Fase de organización.*

El objetivo a obtener en esta fase es la implementación de una solución (arquitectura del sistema, estructura organizativa, limitaciones a tener en cuenta, etc.).

Es importante, durante el desarrollo de esta fase operativa, validar un conjunto de aspectos básicos:

— Organización prevista del trabajo en los servicios involucrados, impacto sobre las organizaciones, etc.;

— estructura general de la organización de los datos en ficheros y bases de datos;

— estructura general del software (programas y aplicaciones) que será necesario preparar;

— limitaciones y condiciones de trabajo impuestas a (o desde) los servicios de explotación informática (seguridad, horarios limitados en teletratamiento, fechas, etc.).

d) *Fase de validación.*

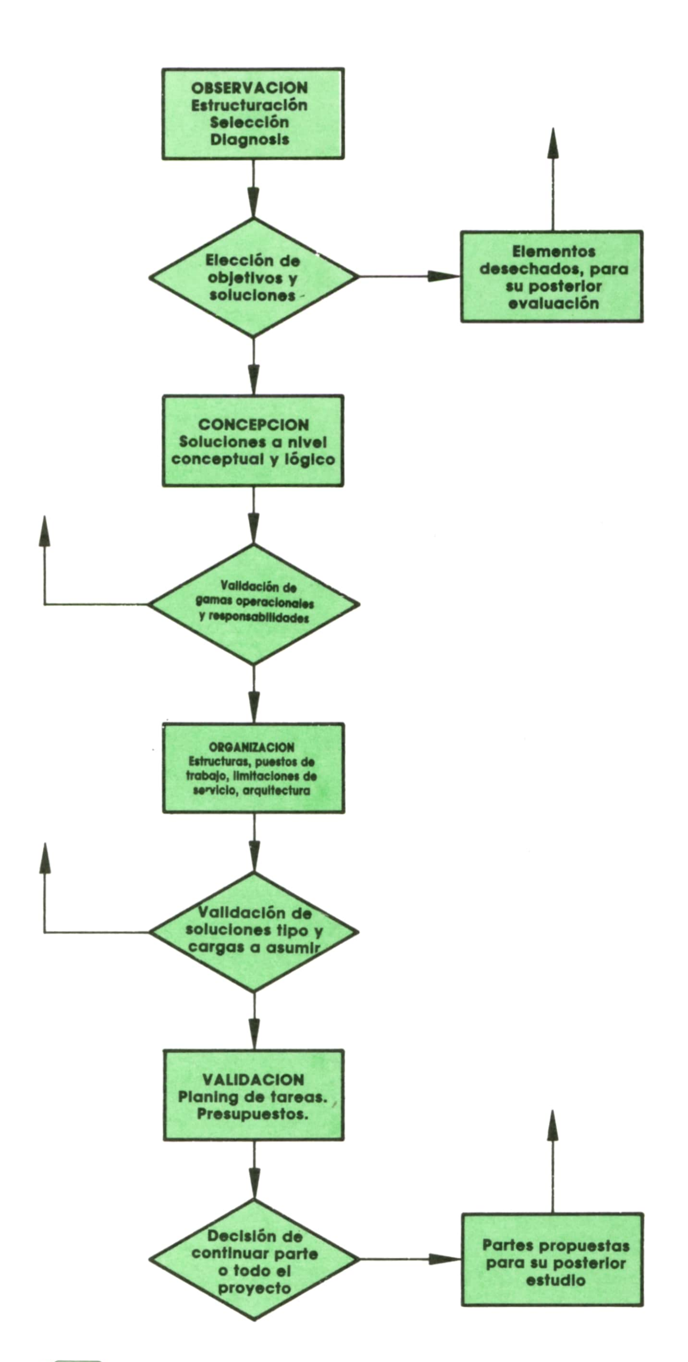
Se establecen los planings de trabajo y sus correspondientes presupuestos así como se organizan las tareas a realizar. Con el desarrollo de esta fase y los presupuestos económicos detallados que se elaboren, el comité director debe tomar su decisión respecto de:

— la aceptación final de todas o parte de las soluciones propuestas;

— la descomposición en subproyectos consecutivos o paralelos de desarrollo de aplicaciones;

— el planing de los estudios detallados, de las realizaciones y puestas en marcha correspondientes;

— la aceptación de los presupuestos correspondientes a los subproyectos y tareas aceptados.





Ciclo del «Estudio de viabilidad».



TECNICAS DE PROGRAMACION

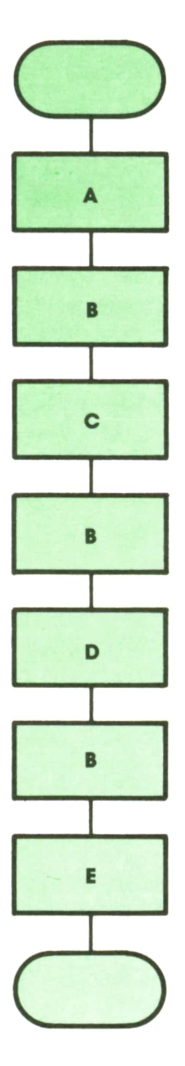


Programación modular (continuación)

EN el lenguaje BASIC es recomendable hacer el mayor uso posible de la programación modular, al menos en la escasa proporción en que este lenguaje la permite. Para ello, todo programa debe distinguir claramente lo que podríamos llamar el «programa principal» de todas aquellas partes (subprogramas o módulos) cuya ejecución puede desencadenarse desde distintas partes del programa, o incluso de aquellas otras que, aun no utilizándose más que en un solo punto, pudiera ser conveniente separar, ya sea porque constituyen un todo cerrado o por otras razones semejantes.

Naturalmente, siempre es posible prescindir de la programación modular y construir nuevos programas, haciendo uso exclusivo de las instrucciones de control que hemos visto en capítulos anteriores. Sin embargo, esto no es aconsejable, porque dificulta la comprensión de los programas. Veamos un ejemplo.

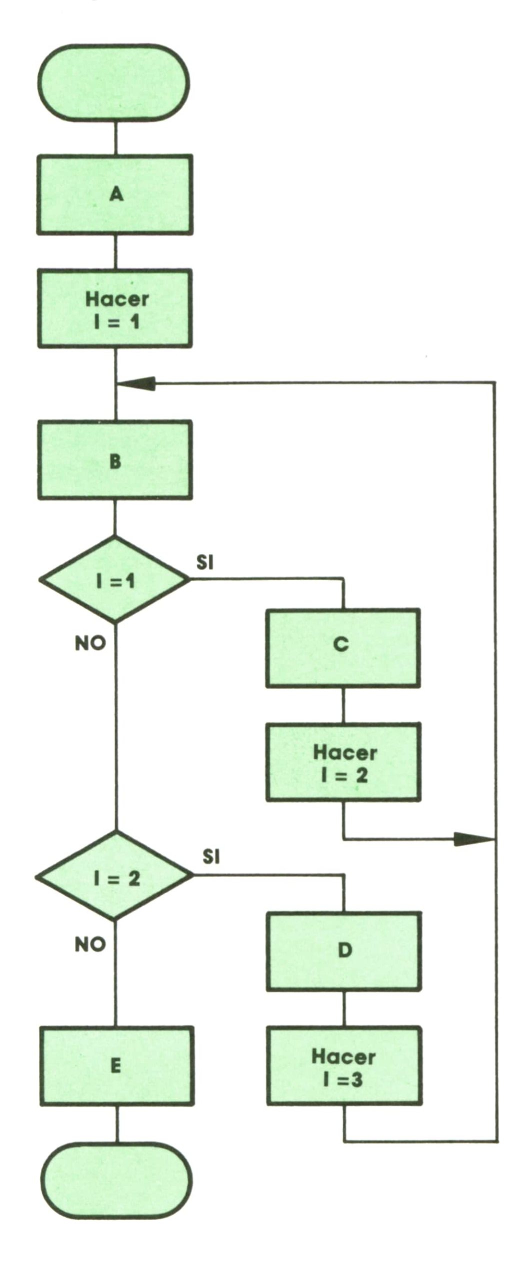
Supongamos que tenemos el programa BASIC representado por el organigrama siguiente:



donde cada uno de los bloques rectangulares no representa una sola instrucción, como en organigramas anteriores, sino conjuntos de instrucciones que lla-

maremos con las letras A, B, C, D y E, y que pueden contener bucles, instrucciones condicionales, etc. Se observará que el conjunto de instrucciones B está repetido en nuestro programa en tres sitios diferentes. Como es natural, sería una pena (y un gasto innecesario de memoria y de mecanografía) escribirlo tres veces completas. Por tanto, debemos buscar una manera de conseguir escribirlo una sola vez.

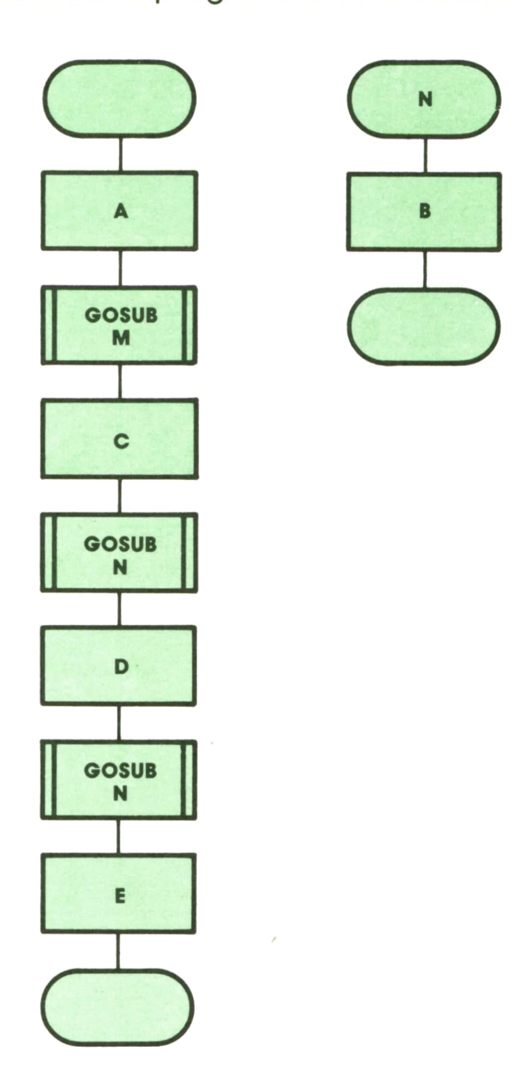
La primera solución que podría ocurrírsenos viene representada por el organigrama siguiente:



donde se supone que la variable I es nueva, que no tomaba parte en ninguna de las secciones del programa anterior.

Esta solución no es aceptable. Es verdad que se consigue lo que se deseaba (B aparece una sola vez), pero a costa de complicar excesivamente la estructura del conjunto. Además, imagínese que existen otras secciones del programa a las que se pueda aplicar, asimismo, esta reducción. El organigrama se complicaría entonces hasta extremos inadmisibles.

Veamos cuál es la solución correcta, utilizando la programación modular:



Se observará que la estructura del programa principal no se complica en este caso. Que cada bloque B ha sido sustituido por una sola instrucción (GOSUB N, donde N es una etiqueta nueva, adjudicada al módulo) y que el conjunto B aparece una sola vez, separado del programa principal. Basta comparar esta figura con la anterior para darse cuenta de la ventaja de este método para mejorar la

facilidad de comprensión de nuestros programas. Además, si la situación se repitiera y nos viéramos en la necesidad de separar otro módulo, la estructura de nuestro programa no se complicaría.

Otra razón que puede movernos a separar un módulo o subrutina, aun en el caso de que sólo se utilice una vez, es que dicho módulo realice una función que puede repetirse en un programa totalmente diferente. En tal caso, sería bueno tener programados todos los módulos de esta clase en un conjunto único, que utilizaremos como base cada vez que vamos a construir un programa nuevo, y en cada caso nos limitaremos a eliminar los que no nos son necesarios para la aplicación concreta que estamos programando, añadiendo después el programa principal correspondiente. Por esta razón, es conveniente asignar a los módulos etiquetas muy altas, para que no interfieran con las de los programas principales que nos puedan surgir.

Entre las operaciones que puede tener sentido separar de todos los programas principales para constituir nuestra biblioteca de módulos, podemos citar las siguientes:

— Aguardar a que se presione una tecla y devolver en una variable de caracteres el carácter correspondiente.

— Aguardar a que se presione una tecla de función, ignorar las restantes y devolver en una variable numérica el número de la función correspondiente.

— Pedir un dato numérico al teclado, comprobar que está dentro de ciertos límites y devolver en una variable numérica el valor obtenido. En caso contrario, escribir un mensaje de error apropiado y volver a pedir el dato.

— Borrar áreas determinadas de la pantalla.

— Guardar en un fichero en disco los valores de ciertas variables.

— Recuperar de un fichero en disco los valores de ciertas variables.

— Realización de ciertas operaciones aritméticas muy frecuentes.

Se observará que se trata, casi siempre, de operaciones de entrada o salida de datos por dispositivos periféricos. Volveremos sobre esto en capítulos posteriores.

En el tomo primero vimos un programa de animación muy sencillo, escrito en BASIC, que dibujaba en la pantalla un recipiente con dos electrodos, de los que salían burbujas móviles ascendentes. Vamos a ver ahora una versión más completa de este mismo programa, que utiliza la programación modular.

```
5 REM Borrar la pantalla
10 CLS
15 REM Dibujar un recipiente lleno de agua
20 LOCATE 10,20:PRINT CHR$(191)+SPACE$(20)+CHR$(218)
30 LOCATE 11,20:PRINT CHR$(179)+SPACE$(20)+CHR$(179)
40 LOCATE 12,20:PRINT CHR$(179)+SPACE$(20)+CHR$(179)
50 LOCATE 13,20:PRINT CHR$(195)+STRING$(20,196)+CHR$(180)
60 LOCATE 14,20:PRINT CHR$(179)+SPACE$(20)+CHR$(179)
70 LOCATE 15,20:PRINT CHR$(179)+SPACE$(20)+CHR$(179)
80 LOCATE 16,20:PRINT CHR$(179)+SPACE$(20)+CHR$(179)
90 LOCATE 17,20:PRINT CHR$(179)+SPACE$(20)+CHR$(179)
100 LOCATE 18,20:PRINT CHR$(192)+STRING$(20,196)+CHR$(217)
105 REM Dibujar las opciones a elegir
106 REM 1: Un tubo cerrado, invertido y sumergido en el líquido
107 REM 2: Un tubo abierto sumergido en el líquido
108 REM 3: Una jeringuilla (animación)
109 REM 0: Fin del juego
110 LET X=4:LET Y=50:GOSUB 1000
120 LET X=4:LET Y=60:GOSUB 1200
130 LET X=4:LET Y=70:GOSUB 1200:GOSUB 1400
190 LOCATE 12,53: PRINT"1         2         3"
199 REM Elegir opción
200 LOCATE 1,1
205 PRINT"Elige un número (1,2,3,0) presionando la tecla correspondiente"
209 REM Leer un número (1,2,3,0) del teclado
210 LET A$=INKEY$:IF A$="" THEN 210
215 REM Comprobar que la opción elegida es correcta
220 IF A$<>"1" AND A$<>"2" AND A$<>"3" AND A$<>"0" THEN 210
225 REM Transferir control, según la opción elegida
230 ON ASC(A$)-47 GOTO 999,300,400,500
299 REM Primera opción: tubo cerrado
300 LET X=4:LET Y=50:GOSUB 1600
```

```
310 LET X=10:LET Y=28:GOSUB 1000
320 LOCATE 16,29:PRINT STRING$(4,196)
330 GOTO 900
399 REM Segunda opción: tubo abierto
400 LET X=4:LET Y=60:GOSUB 1600
410 LET X=10:LET Y=28:GOSUB 1200
420 LOCATE 13,29:PRINT STRING$(4,196)
430 GOTO 900
499 REM Tercera opción: jeringuilla
500 LET X=4:LET Y=70:GOSUB 1600
510 LET X=10:LET Y=28:GOSUB 1200:GOSUB 1400
520 LOCATE 16,29:PRINT STRING$(4,196):LET K=0:LET K1=1+RND*5:LET Z=16
530 FOR J=1 TO K1:GOSUB 1900: GOSUB 1800: LET X=X+1: GOSUB 1400:GOSUB 2000
535 NEXT J
540 LET K=7-K1
550 FOR J=1 TO K1:GOSUB 1900: GOSUB 1800: LET X=X-1: LET Z=Z-1: GOSUB 1400
560 LOCATE Z,Y+1: PRINT STRING$(4,196): NEXT J
899 REM Opción terminada. ¿Continuamos?
900 LOCATE 1,1:PRINT SPACE$(80):LOCATE 1,1:PRINT "¿Otra vez? (s/n)"
910 LET A$=INKEY$: IF A$="" THEN 910
920 IF A$<>"s" AND A$<>"n" THEN 910
930 IF A$="s" THEN 10
999 END
1000 REM Pintar el tubo cerrado
1010 LOCATE X,Y:PRINT CHR$(218)+STRING$(4,196)+CHR$(191)
1020 LOCATE X+1,Y:PRINT CHR$(179)+SPACE$(4)+CHR$(179)
1030 LOCATE X+2,Y:PRINT CHR$(179)+SPACE$(4)+CHR$(179)
1040 LOCATE X+3,Y:PRINT CHR$(179)+SPACE$(4)+CHR$(179)
1050 LOCATE X+4,Y:PRINT CHR$(179)+SPACE$(4)+CHR$(179)
1060 LOCATE X+5,Y:PRINT CHR$(179)+SPACE$(4)+CHR$(179)
1070 LOCATE X+6,Y:PRINT CHR$(217)+SPACE$(4)+CHR$(192)
1080 RETURN
1200 REM Pintar el tubo abierto
1210 LOCATE X,Y:PRINT CHR$(191)+SPACE$(4)+CHR$(218)
1220 LOCATE X+1,Y:PRINT CHR$(179)+SPACE$(4)+CHR$(179)
1230 LOCATE X+2,Y:PRINT CHR$(179)+SPACE$(4)+CHR$(179)
1240 LOCATE X+3,Y:PRINT CHR$(179)+SPACE$(4)+CHR$(179)
1250 LOCATE X+4,Y:PRINT CHR$(179)+SPACE$(4)+CHR$(179)
1260 LOCATE X+5,Y:PRINT CHR$(179)+SPACE$(4)+CHR$(179)
1270 LOCATE X+6,Y:PRINT CHR$(217)+SPACE$(4)+CHR$(192)
1280 RETURN
1400 REM Pintar la jeringuilla
1405 LOCATE X-1,Y+2:PRINT CHR$(218)+CHR$(191)
1410 LOCATE X,Y+1:PRINT CHR$(218)+CHR$(193)+CHR$(193)+CHR$(191)
1420 RETURN
1600 REM Borrar un tubo
1605 FOR I=1 TO 8
1610 LOCATE X+I-2,Y:PRINT SPACE$(10)
1620 NEXT I
1630 RETURN
1800 REM Borrar la jeringuilla
1805 LOCATE X-1,Y+2:PRINT SPACE$(2)
1810 LOCATE X,Y+1:PRINT SPACE$(4)
1815 FOR I=1 TO K:LOCATE X+I,Y+1:PRINT SPACE$(4):NEXT I
1820 RETURN
1900 REM Retardo
1910 FOR I=1 TO 100:A=2.5*2.5:NEXT I:RETURN
2000 REM Burbujitas
2010 X1=17:Y1=27
2020 FOR I=1 TO 3:X1=X1-1:LOCATE X1,Y1: PRINT "o":LOCATE X1,Y1+7:PRINT"o":NEXT
2025 X1=17
2030 FOR I=1 TO 3:X1=X1-1:LOCATE X1,Y1: PRINT " ":LOCATE X1,Y1+7:PRINT" ":NEXT
2090 RETURN
```

El programa dibuja una cubeta llena de agua, un tubo abierto por un extremo y cerrado por el otro, un tubo abierto por los dos extremos y una jeringuilla. Se tiene entonces la posibilidad de elegir uno de estos tres tubos, hundirlo parcialmente en la cubeta y ver qué ocurre con el nivel del agua en el interior del tubo correspondiente. En el caso de la jeringuilla se produce también un pequeño efecto de animación, que hace ver cómo se expulsa el aire del interior del

tubo y cómo se aspira el líquido al subir de nuevo el émbolo.

No vamos a explicar con detalle la construcción del programa, pues los muchos comentarios incluidos la harán evidente. Obsérvese, sin embargo, que este programa contiene los siete módulos o subrutinas siguientes:

— **Etiqueta 1000.** Dibuja un tubo cerrado en la posición de la pantalla definida por las variables X (línea) e Y (columna). Este módulo se utiliza dos veces a lo largo del programa.

— **Etiqueta 1200.** Dibuja un tubo abierto en la posición de la pantalla definida por las variables X (línea) e Y (columna). Este módulo se utiliza cuatro veces a lo largo del programa.

— **Etiqueta 1400.** Dibuja el émbolo de la jeringuilla en la posición de la pantalla definida por las variables X (línea) e Y (columna). Este módulo se utiliza cuatro veces a lo largo del programa.

— **Etiqueta 1600.** Borra un tubo de la posición de la pantalla definida por las variables X (línea) e Y (columna). Este módulo se utiliza tres veces a lo largo del programa.

— **Etiqueta 1800.** Borra el émbolo de la jeringuilla de la posición de la pantalla definida por las variables X (línea) e Y (columna). Este módulo se utiliza dos veces a lo largo del programa.

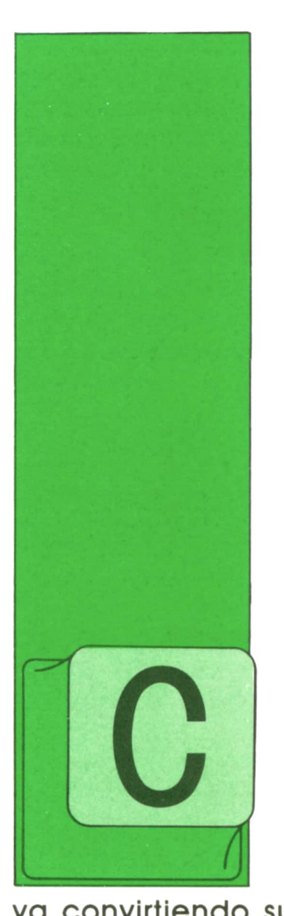
— **Etiqueta 1900.** Utilizado durante la animación, produce un retardo o tiempo muerto para que el movimiento no sea excesivamente rápido. Este módulo se utiliza dos veces a lo largo del programa.

— **Etiqueta 2000.** Dibuja las burbujas que salen de la jeringuilla al presionar el émbolo. El movimiento de estas burbujas constituye la animación propia de este programa. Este módulo se utiliza una sola vez a lo largo del programa (dentro de un bucle), pero era conveniente separarlo, pues realiza una acción independiente y completa en sí misma.

El programa anterior está escrito para IBM PC y compatibles. Sin embargo, los cambios indicados en la versión del tomo 1 para AMSTRAD, COMMODORE y MSX son también aplicables aquí, sin demasiado esfuerzo.

APLICACIONES

INTRODUCCION A LAS BASES DE DATOS



CUALQUIER organización moderna depende de sus recursos de procesos de datos tanto como del resto de sus componentes. En la medida en que una empresa va convirtiendo sus tareas manuales en aplicaciones para ordenador, pasa a ser más eficiente y productiva. Esta relación está basada esencialmente en los datos que soportan y manejan estas aplicaciones.

Proporcionar el acceso a los datos que una organización genera y emplea supone una continua y significativa inversión en desarrollo de programas, mantenimiento, equipos de proceso y almacenamiento de datos, terminales de usuario y servicios de comunicación. Es necesario tomar medidas adecuadas para conseguir que el acceso a los datos sea sencillo, rápido y controlable para asegurar un uso correcto de los datos de la organización. La combinación del hardware, el software y los procedimientos para dejar los datos preparados para ser utilizados se denominan normalmente: ENTORNO DEL SISTEMA DE DATOS.

La forma más normal de abordar el tratamiento de datos es mediante el uso de bases de datos. Es, como siempre en informática, difícil encontrar una definición universal de base de datos. Sin embargo, el concepto descrito por todas las definiciones es muy similar. Una base de datos puede considerarse:

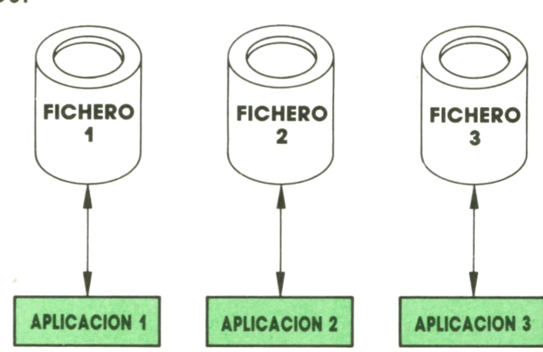
«El conjunto de datos no redundantes, interrelacionados, con posibilidad de ser procesados por una o más aplicaciones.»

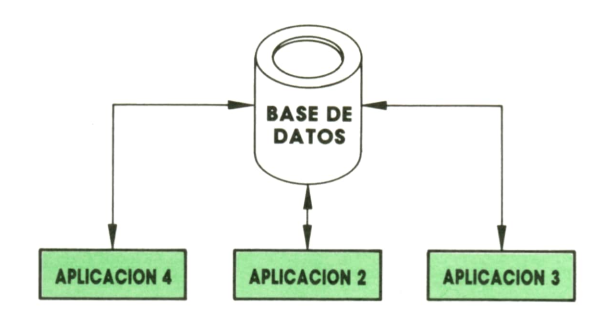
Esta definición formal contiene la esencial del concepto de base de datos, frente a las características de fichero convencional:

— *No redundantes.* Significa el mínimo número de elementos para proporcionar la entrada a los programas de aplicación. Como consecuencia de ello el número de un cliente sólo figurará una vez en la base de datos, aunque todos los programas de aplicación tengan que acceder a ese dato.

— *Interrelacionados.* Supone que los datos de unos campos están relacionados con otros por apuntadores, tablas o claves.

— *Procesables por varias aplicaciones.* Se refiere al entorno informático actual en que cientos de usuarios en diferentes terminales necesitan acceder a datos comunes para propósitos diferentes.





Evolución histórica

En un principio, los registros de datos se agrupaban en ficheros y estos ficheros eran creados y mantenidos por cada aplicación que se ejecutaba en el ordenador. Los contenidos de un fichero y su mantenimiento eran responsabilidad del grupo de usuarios por los que había sido desarrollada la aplicación. Cuando la mayor parte del trabajo realizado era manual y el uso del ordenador era pequeño, esta forma de funcionar proporcionaba buenos resultados. A medida que fue creciendo el número de datos a manejar y el número de aplicaciones que los usaban aparecieron problemas causados por esta metodología. En ella se disponía de ficheros por aplicación y por ello:

— Aparecían datos duplicados en diferentes ficheros.

— Los contenidos de estos datos redundantes podrían ser distintos de un fichero a otro. Ello generaba la posibilidad de falta de consistencia en los datos del sistema.

— La tarea de resumir y consolidar los datos para efectuar tareas de gestión se dificultaba enormemente a media que el número de ficheros crecía.

— Los datos y los programas eran enormemente dependientes. Un cambio en el formato de un campo suponía rehacer y actualizar todos los programas implicados.

— Algunos programas incluso estaban diseñados para dispositivos físicos específicos, y al cambiar éstos por otros más modernos, los programas dejaban de funcionar.

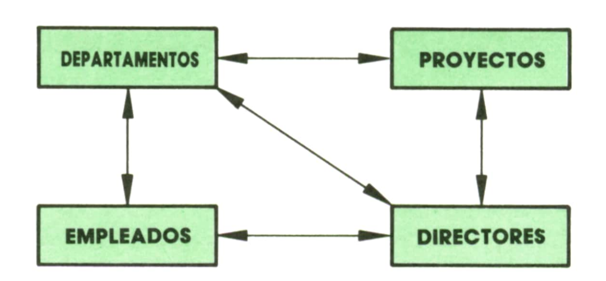
Todas estas condiciones hacen que los programas de aplicación estén en constante cambio y se desperdicien recursos, tanto económicos, personales como de almacenamiento. La solución a los problemas generados por la existencia de múltiples ficheros de datos es combinar estos ficheros de datos en bases de datos comunes.



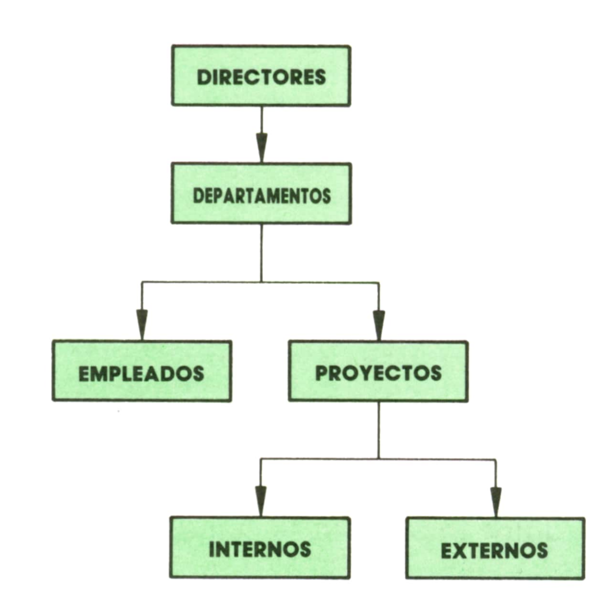
Bases de datos

Existen tres formas fundamentales de establecer las relaciones entre los datos de una base de datos. Estos tres modelos determinan el tipo de base de datos y su funcionamiento.

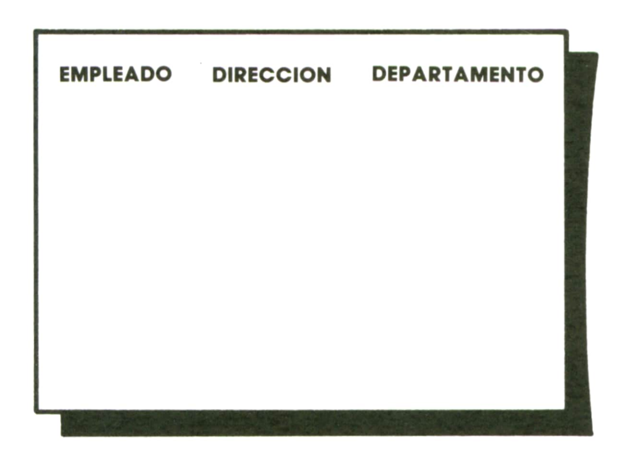
— En primer lugar, el modelo en red que mantiene una estructura de malla entre los datos con apuntadores entre los campos de los registros.



— Un segundo modelo es el jerárquico, donde las relaciones entre registros mantienen una dependencia «padre-hijo».



— Por último, las de más reciente aparición y de mayor potencia son las bases de datos relacionales. Estas basan su estructura y funcionamiento en el mantenimiento de tablas y relaciones entre los datos. Las consultas se efectúan aplicando funciones básicas sobre los datos, y con ellas se generan nuevas tablas, que son los resultados. La estructura elemental de trabajo es la relación entre datos por tablas.



La forma de crear bases de datos consiste en definir con claridad los campos de que consta cada registro e identificar las relaciones que existen entre éstos. En el caso jerárquico se establece una estructura de tipo arbórea, que refleja las dependencias de un registro con sus «hijos», etc. Los datos se relacionan por apuntadores que «señalan» a sus correspondientes relacionados. Los datos están representados tal y como los ven los programadores y no como se encuentran físicamente en el disco. Se generan los segmentos de datos y se establecen las relaciones de parentesco. Cuando un registro no tiene padre se le considera el registro raíz y contiene la clave que identifica un registro que es función de todos sus descendientes.

En el caso de las bases de datos relacionales, los datos aparecen en forma tabular y los usuarios pueden tener acceso a ellos desde distintas vistas. Existen algunas reglas básicas que definen si una base de datos es totalmente relacional o no lo es. De entre ellas destacan:

1. Toda la información de una base de datos relacional se representa en el nivel lógico de una única manera: por valores en tablas.

2. Todos y cada uno de los datos de una base de datos relacional deben tener la garantía de ser accesibles mediante la combinación de un nombre de tabla, una clave primaria y un nombre de columna.

 Consideraciones adicionales

Una vez introducido el concepto de base de datos, y comentados brevemente algunos de los tipos básicos, pasamos a citar algunos puntos interesantes sobre las bases de datos.

Como ya hemos comentado, no existe una definición ampliamente aceptada de lo que es una base de datos. Del mismo modo tampoco está claramente definida la estrategia a seguir a la hora de desarrollar una base de datos. En este sentido, el concepto de base de datos guarda una estrecha relación con la noción de sistema de gestión de información. Es decir, considerar la información como un bien más de la empresa. Ello obliga a desarrollar bases de datos que permitan a los usuarios acceder a esos datos y facilitar vías de acceso rápidas para las aplicaciones. Esta forma de concebir una base de datos, se ha revelado muy difícil de realizar en la práctica. Algunos de los problemas no están originados por la técnica, sino por la estructura organizativa:

— Problemas organizativos, debidos a la forma de compartir los datos.

— Los causados por la necesidad de incorporar un nuevo papel en la empresa: el administrador de bases de datos. Además, es necesario conseguir que esta figura quede perfectamente integrada en la estructura ya existente.

— La carencia de normas y procedimientos estándares eficaces para la gestión y el manejo de los datos.

— La incapacidad de las empresas para prever a largo plazo la evolución de su «arsennal» de datos.

— La negativa de los directivos a invertir a largo plazo, cosa que las bases de datos requieren.

Al margen de toda esta problemática, cuando se plantea el análisis y el diseño de una base de datos es esencial que se determine con gran exactitud para qué

se necesita esa base de datos. Es necesario tener en cuenta que una base de datos no es otra cosa que una parcela del mundo real que vamos a representar mediante conjuntos de caracteres. Esta fase de definición es crítica a la hora de preparar la base de datos. Debe tenerse en cuenta qué datos son los verdaderamente necesarios para describir esa parte de la realidad que queremos identificar con exactitud.

Normalmente, incluso los mismos usuarios que participan en el proceso no saben discernir si vale o no la pena incluir determinada información, o si es técnica o económicamente viable incorporarla a la base de datos. Para resolver este conflicto es necesario reconocer que una cosa es la planificación y otra muy distinta la construcción práctica del sistema. La construcción se realiza de forma gradual. Una vez tomadas las elecciones de las aplicaciones en concreto, aparecerá la necesidad de los datos. Por ello, corresponde al diseño (modelo de datos) de la base de datos la misión de aportar el marco unificado que garantice que todo nuevo dato encajará perfectamente en el conjunto.

Una posible orientación a seguir es, pues, incluir en el plan general (es decir, en el diseño del modelo de datos) todo cuanto pueda resultar de utilidad, pero cuidar de justificar su inclusión en la base de datos real.

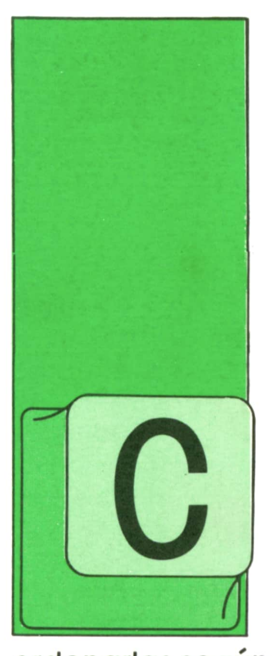
La consideración de los objetivos comerciales estratégicos de la compañía y de la repercusión que éstos puedan tener sobre las diversas actividades de la firma es de gran ayuda para calibrar qué aspectos pueden resultar importantes y cuáles no en una determinada base de datos. Es más, si se llega a relacionar determinados elementos del proyecto o modelo de datos con aspectos de la estrategia comercial, se estará en óptimas condiciones para entablar diálogo con la dirección.



Resumen

Como resumen podemos indicar que la construcción de las bases de datos como un modelo a escala de la realidad nos ayuda a centrar la atención en algunos de los objetivos más importantes de una base de datos. Uno de ellos es el de permitir manejar los datos como si de un bien se tratara. Por ello, es importante conservar los mismos datos para utilizarlo en nuevos propósitos, sin necesidad de construir nuevas bases de datos. Esta idea debiera hacer replantearse la cuestión a los responsables de gestión de información, tendiendo a cuidar más el mantenimiento y gestión de los datos que la aparición de nuevas aplicaciones.

PASCAL



Listas encadenadas (continuación)

CUANDO introdujimos el concepto de almacenamiento dinámico, mencionamos como ejemplo los casos en que hay que guardar en memoria muchas fichas para ordenarlas según un criterio dado y no se sabe a priori el número de éstas con que se va a trabajar.

Con las operaciones básicas de manejo de listas encadenadas que hemos visto, escribir un programa de ordenación por el método de selección directa sería muy sencillo: bastaría, en primer lugar, con ir leyendo los datos de cada ficha y añadirla a una lista; el procedimiento de ordenación de la lista, propiamente dicho, sería muy parecido al de una tabla, pues lo que, en la ordenación de éstas, eran los dos índices empleados para recorrerlas, aquí serían dos punteros que irían recorriendo la lista.

Veamos cómo podrían ser los procedimientos para leer y mostrar los datos de las diferentes fichas.

```
program CreaLista;

const
  Long = 12;    (* Máximo número de letras por nombre *)

type
  Nombre_t = array [1..Long] of char;
  Punt_t   = ^Ficha_t;
  Ficha_t = record
              Nombre,
              Apellido1,
              Apellido2  : Nombre_t;
              Nota_A,
              Nota_B,
              Nota_C     : real;
              Siguiente  : Punt_t
            end;

var
  Primera   : Punt_t;
(*-----------------------------------------------------------------*)
procedure LeeDatos (var Primi: Punt_t);

(*  Pide datos y los guarda en una lista  *)
(*  que queda apuntada por Primi.         *)

  var
   P    : Punt_t;
   N    : Nombre_t;
   Fin  : boolean;

begin
 writeln ('Para acabar, introduzca como nombre 0.');
 writeln ('Comience a introducir datos.');

 Primi := nil;                    (* Crea una lista vacía. *)

 repeat
   writeln;
   write ('Nombre:                 '); readln (N);
   Fin := (N[1] = '0');
   if not Fin then
     begin
      new (P);                    (* Crea una nueva ficha. *)
      with P^ do                  (* Rellena la ficha.     *)
        begin
```

```
            Nombre := N;
            write ('Primer apellido:  '); readln (Apellido1);
            write ('Segundo apellido: '); readln (Apellido2);
            writeln;
            write ('Nota A: '); readln (Nota_A);
            write ('Nota B: '); readln (Nota_B);
            write ('Nota C: '); readln (Nota_C);

            Siguiente := Primi;     (* Añade la ficha a la lista. *)
            Primi := P
           end
        end
 until Fin
end;

(*-----------------------------------------------------------------*)
procedure Presenta (Primi: Punt_t);

(* Presenta el contenido de la lista apuntada por Primi. *)

var Actual: Punt_t;

begin
 Actual := Primi;

 while Actual <> nil do with Actual^ do
   begin
    writeln;
    writeln (Nombre,' ',Apellido1,' ',Apellido2);
    writeln ('Nota A: ',Nota_A :4:1);
    writeln ('Nota B: ',Nota_B :4:1);
    writeln ('Nota C: ',Nota_C :4:1);

    Actual := Siguiente
   end
end;

(*-----------------------------------------------------------------*)
begin
 LeeDatos (Primera);
 Presenta (Primera)
end.
```

Listas ordenadas

Si se quisiera insertar un nuevo elemento en medio de una tabla, habría que hacer sitio primero, desplazar todos los elementos posteriores un lugar hacia atrás y almacenar aquél en la tabla; a medida que la tabla y la cantidad de información contenida en cada ficha se fuese haciendo mayor, el tiempo que necesitaría el ordenador se iría incrementando.

Insertar un elemento en una lista es cuestión de retocar unos pocos punteros y poco más, la cantidad de operaciones necesarias es independiente del tamaño de la lista. El problema de leer unas fichas y ordenarlas se puede resolver de una manera mucho más eficiente si cada vez que se lean los datos de una ficha nueva, en lugar de añadirla por delante de la lista, se inserta en el lugar que le corresponda directamente. El procedimiento LeeDatos sería:

```
procedure LeeDatos (var Primi: Punt_t);

(*  Pide datos y los guarda en una lista    *)
(*  ordenada que queda apuntada por Primi.  *)

  var
   P   : Punt_t;
   N   : Nombre_t;
   Fin : boolean;
  (*-----------------------------------------------*)
  procedure Insertar;
     (* Coloca la ficha P^ en el sitio correspondiente *)
     (* de la lista apuntada por el puntero Primi.     *)

   begin
     (*   ESTA DESCRITO EN SITIO APARTE, VEASE EL TEXTO     *)
   end;
(*-------------------------------------------------*)
```

```
begin
 writeln ('Para acabar, introduzca como nombre 0.');
 writeln ('Comience a introducir datos.');

 Primi := nil;                       (* Crea una lista vacía. *)

 repeat
   writeln;
   write ('Nombre:               '); readln (N);
   Fin := (N[1] = '0');
   if not Fin then
     begin
      new (P);                       (* Crea una nueva ficha. *)
      with P^ do );                  (* Rellena la ficha.     *)
        begin
         Nombre := N;
         write ('Primer apellido:  '); readln (Apellido1);
         write ('Segundo apellido: '); readln (Apellido2);
         writeln;
         write ('Nota A: '); readln (Nota_A);
         write ('Nota B: '); readln (Nota_B);
         write ('Nota C: '); readln (Nota_C)
        end;

      Insertar                       (* Inserta la ficha P^ *)
     end
 until Fin
end;
```

El procedimiento Insertar debe recorrer la lista hasta encontrar el elemento que deba situarse por detrás del que queremos añadir para, a continuación, insertar éste por delante.

```
procedure Insertar;
   (* Coloca la ficha P^ en el sitio correspondiente *)
   (* de la lista apuntada por el puntero Primi.     *)
  var
    Actual  : Punt_t;
    EsMenor: boolean;
    Auxi    : Ficha_t;

 begin
 (*-------------------------------------------------------*)
 (* Recorrer la lista hasta encontrar la primera ficha    *)
 (* que deba ir por detrás de P^ o hasta llegar al final: *)
 (*-------------------------------------------------------*)

 Actual  := Primi;
 EsMenor:= false;

 while (Actual <> nil) and not EsMenor do
   if  Actual^.Nota_A < P^.Nota_A then EsMenor := true
       else  Actual := Actual^.Siguiente;

 (*-------------------------------------------------------*)
 (* Añadir P^ por delante de la ficha encontrada o al     *)
 (* final si no existe:                                   *)
 (*-------------------------------------------------------*)

 if EsMenor then        (* insertar por delante de Actual^ *)
     begin
      Auxi    := P^;
      P^      := Actual^;
      Auxi.Siguiente := P;
      Actual^:= Auxi
     end
   else
     begin                          (* insertar tras la última  *)

       (*   ¡¡ VEASE DE NUEVO EL TEXTO !!   *)

     end

 end;
```

Si no hay ninguna ficha que deba quedar por detrás de la nueva, ésta debemos ponerla detrás de la hasta entonces última, pero como esa situación se detecta cuando el puntero Actual vale NIL, ya no es posible retocar su campo Siguiente. Debemos recorrer la lista guardando en cada momento un puntero que apunte a la ficha anterior a la actual.

Cuando se vaya a introducir la primera ficha de todas, Primi vale NIL, por lo que no se llega a ejecutar ni una vez el bucle WHILE; esta situación se puede detectar luego porque es el único caso en que Previa puede ser NIL.

Una vez esté incluido este procedimiento en LeeDatos, y éste, a su vez, haya reemplazado a su homónimo del programa CreaLista, observaremos que las fichas introducidas se presentan ordenadas según el valor del campo Nota-A sin haber ejecutado realmente un procedimiento de ordenación.

```
procedure Insertar;
   (* Coloca la ficha P^ en el sitio correspondiente *)
   (* de la lista apuntada por el puntero Primi.     *)
  var
    Actual,
    Previa : Punt_t;
    EsMenor: boolean;
    Auxi   : Ficha_t;

 begin

 Actual := Primi;
 Previa := nil;
 EsMenor:= false;

 while (Actual <> nil) and not EsMenor do
   if  Actual^.Nota_A < P^.Nota_A then EsMenor := true
       else
         begin
          Previa := Actual;
          Actual := Actual^.Siguiente
         end;

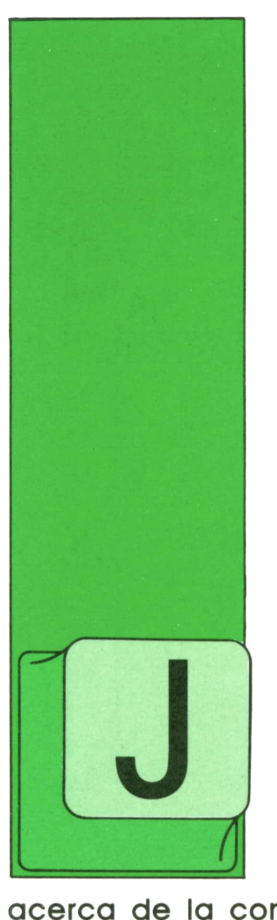
 (*-------------------------------------------------------*)
 (* Añadir P^ por delante de la ficha encontrada o al     *)
 (* final si no existe:                                   *)
 (*-------------------------------------------------------*)

 if EsMenor then        (* insertar por delante de Actual^ *)
     begin
      Auxi   := P^;
      P^     := Actual^;
      Auxi.Siguiente := P;
      Actual^:= Auxi
     end
   else
     if Previa <> nil then     (* insertar tras la última  *)
         begin
          Previa^.Siguiente := P;
          P^.Siguiente := nil
         end
        else                 (* P^ es la primera ficha leida *)
         begin
          Primi := P;
          P^.Siguiente := nil
         end

 end;
```

OTROS LENGUAJES

FORTRAN (y IV)



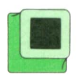


Sentencia GO TO

JUNTO con el BASIC, el FORTRAN es de los pocos lenguajes que disponen de instrucciones de bifurcación. Existe una gran polémica en el mundo de la informática acerca de la conveniencia de los GOTO's frente a algunos aspectos de la programación estructurada. Nosotros no entramos en esa discusión, limitándonos a mostrar el formato de la instrucción:

GO TO num-sentencia

Inmediatamente después el programa continúa su ejecución en la sentencia que tiene el mismo número (columnas 1-5), que el indicado.



Sentencia IF

La sentencia IF permite realizar la bifurcación a otra sentencia del programa siempre que se verifiquen algunas circunstancias especificadas por el programador.

Existen dos formatos distintos de IF: aritmético y condicional. Comenzaremos por este último.

IF condición sentencia

Donde condición es una de las expresiones lógicas explicadas anteriormente, y sentencia es una instrucción cualquiera, excepto un DO (se verá seguidamente) y otro IF.

Si al evaluar la condición, ésta tiene como resultado .TRUE., se ejecutará la sentencia que se encuentra a continuación.

IF expresión aritmética N1 N2 N3

En el formato del IF aritmético. N1, N2 y N3 son números de sentencia que relacionan el IF con otras instrucciones del programa de la siguiente manera: Se calcula el valor de la expresión aritmética. Si éste es menor que cero se bifurcará a la sentencia N1; si es igual a cero, a N2, y si es mayor se ejecutará la instrucción de número N3.

Se puede observar que es equivalente a varios IF lógicos seguidos de GOTO's.



Sentencia DO

Es muy normal que en un mismo programa se deba repetir varias veces un conjunto de sentencias.

En FORTRAN esta iteración o bucle se consigue mediante la sentencia DO, y se conoce al conjunto de sentencias como rango del DO.

```
DO n variable = I1, I2, I3
   rango del DO

n CONTINUE
```

Como se puede observar n es el número de sentencia de la instrucción CONTINUE, cuya única misión es delimitar el

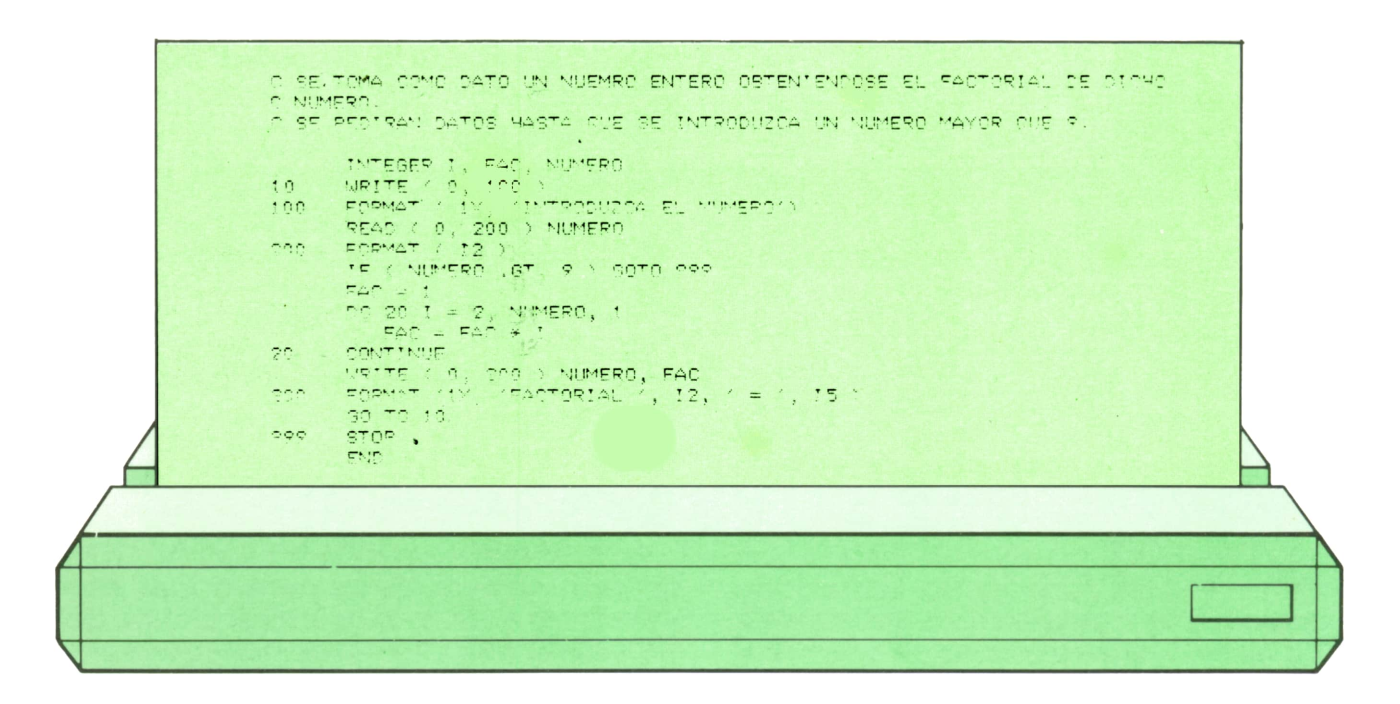
rango del DO, especificar dónde terminan las instrucciones que se deben repetir.

Variable es una variable entera llamada índice del DO.

I1, I2 y I3 pueden ser constantes o variables **enteras.**

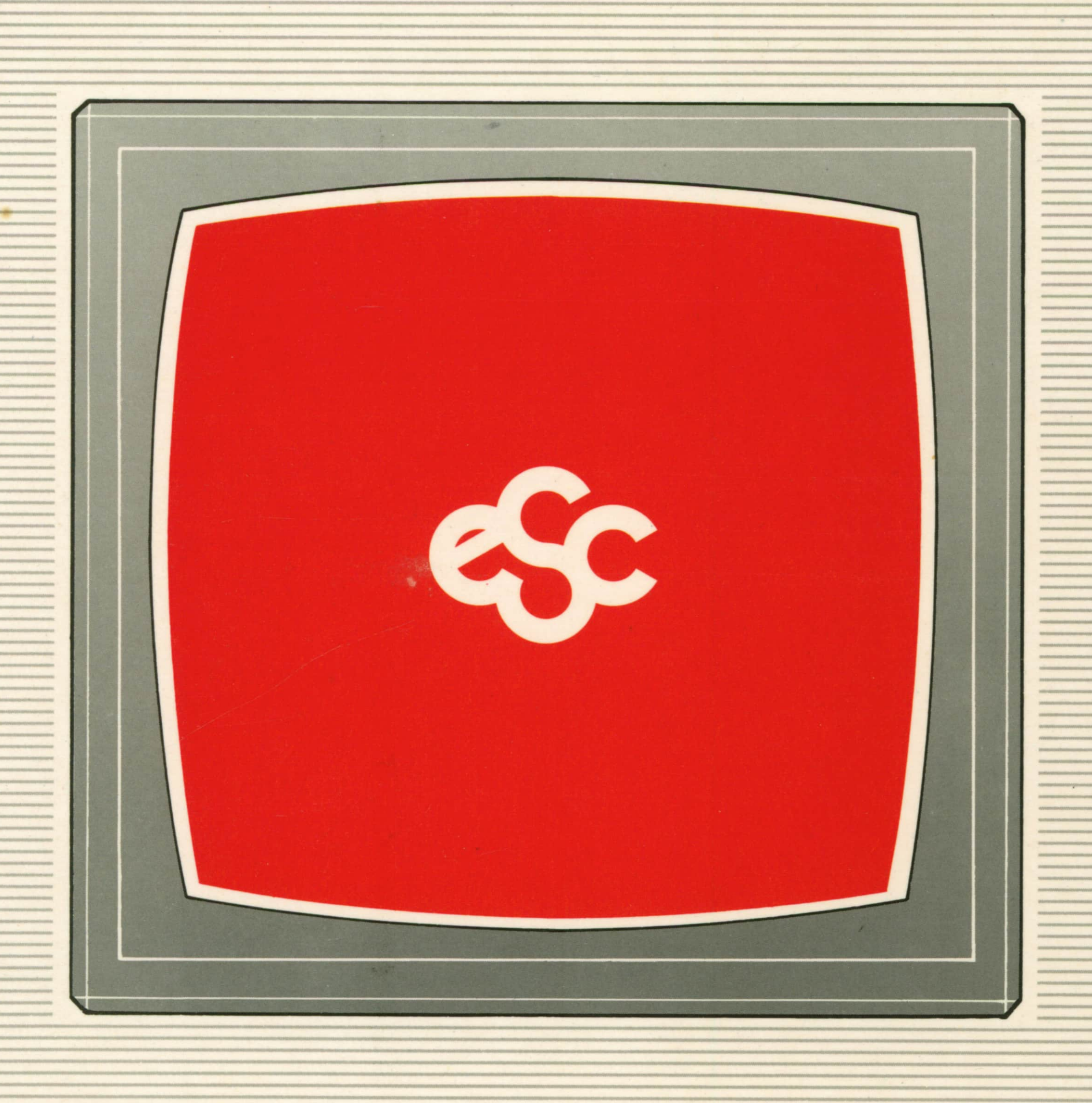
Variable toma primero el valor de I1. Si su valor es mayor que I2, el DO no se ejecuta. Si **variable** es menor o igual que I2 se ejecutan las instrucciones que componen el rango del DO.

Después de la ejecución el valor de **variable** se verá incrementado con el valor de I3 y se vuelve a efectuar la comparación.



```
C SE TOMA COMO DATO UN NUEMRO ENTERO OBTENIENDOSE EL FACTORIAL DE DICHO
C NUMERO.
C SE PEDIRAN DATOS HASTA QUE SE INTRODUZCA UN NUMERO MAYOR QUE 9.

      INTEGER I, FAC, NUMERO
10    WRITE ( 0, 100 )
100   FORMAT ( 1X, 'INTRODUZCA EL NUMERO')
      READ ( 0, 200 ) NUMERO
200   FORMAT ( I2 )
      IF ( NUMERO .GT. 9 ) GOTO 999
      FAC = 1
      DO 20 I = 2, NUMERO, 1
         FAC = FAC * I
20    CONTINUE
      WRITE ( 0, 300 ) NUMERO, FAC
300   FORMAT (1X, 'FACTORIAL ', I2, ' = ', I5 )
      GO TO 10
999   STOP
      END
```



▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼